MASTER IN COMPUTER SCIENCE

Modelling e-business
an approach based on combining UMM and UEML

Riquet, Nicolas

Award date:
2005

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 2004-2005

Modelling e-business: an approach based on combining UMM and UEML

Nicolas Riquet

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique

Abstract

Several languages have been proposed for enterprise modelling and e-business modelling. Each of these modelling languages provides a set of concepts allowing to define models representing certain specific aspects (for instance, the Entities-Associations language defines the concepts of entity, association, etc., to represent data and information about entities). The concepts proposed by the different languages are often partially redundant but slightly different. Therefore, it is often difficult to choose the most adequate language in a given situation. Research efforts are currently being done to attempt to define a unique and common language for enterprise modelling. This language is called UEML (Unified Enterprise Modelling Language) and is the result of the integration of three existing enterprise modelling languages. However, in its current state, UEML only allows to model the enterprise processes and is not able to represent the business collaborations in which the enterprise is involved. This thesis plans to study the feasibility of extending UEML in order to address this lack. To reach this goal, it is necessary to identify the relevant (in terms of business collaboration) concepts to be added to UEML. This identification will be performed through a critical study of UMM (UN/CEFACT Modelling Methodology), which is a good representative of e-business modelling languages. UMM facilitates the analysis of e-business processes and provides reusable process models for the development of e-business platforms. By adding e-business modelling elements to UEML, this thesis intends to provide a language able to model both enterprise processes *and* business collaborations.

Keywords: Modelling language, enterprise modelling, e-business modelling, enterprise process, business collaboration.

Résumé

Plusieurs langages ont été proposés pour la modélisation d'entreprise et la modélisation d'e-business. Chacun de ces langages de modélisation fournit un ensemble de concepts permettant de définir des modèles représentant certains aspects spécifiques (par exemple, le langage Entités-Associations définit les concepts d'entité, d'association, etc., pour représenter des données et des informations à propos des entités). Les concepts proposés par les différents langages sont souvent partiellement redondants mais légèrement différents. Pour cette raison, il est souvent difficile de choisir le langage le plus approprié dans une situation donnée. Des travaux de recherche sont actuellement réalisés pour définir un langage unique et commun pour la modélisation d'entreprise. Ce langage est appelé UEML (Unified Enterprise Modelling Language – Langage Unifié de Modélisation d'Entreprise) et est le résultat de l'intégration de trois langages de modélisation d'entreprise existants. Cependant, dans son état actuel, UEML permet seulement de modéliser les processus d'entreprise et ne permet pas de représenter les collaborations d'affaires dans lesquelles l'entreprise est impliquée. Ce mémoire a pour but d'étudier la faisabilité d'étendre UEML pour combler ce manque. Pour ce faire, il est nécessaire d'identifier les concepts pertinents (en termes de collaboration d'affaires) devant être ajoutés à UEML. Cette identification se fera à travers une étude critique de UMM (UN/CEFACT Modelling Methodology – Méthodologie de Modélisation de l'UN/CEFACT), qui est un bon représentant des langages de modélisation d'e-business. UMM facilite l'analyse des processus d'e-business et fournit des modèles de processus réutilisables pour le développement de plateformes e-business. En ajoutant des éléments de modélisation d'e-business à UEML, ce mémoire a pour but de fournir un langage capable de modéliser à la fois les processus d'entreprise *et* les collaborations d'affaires.

Mots-clés: Langage de modélisation, modélisation d'entreprise, modélisation d'e-business, processus d'entreprise, collaboration d'affaires.

Remerciements (Acknowledgments)

This part of the thesis is written in French because the people I want to thank do not all understand English. In the following paragraphs, I thank my parents, my family and my friends for their support. I also thank Prof. Michaël Petit (University of Namur) and Prof. Giuseppe Berio (University of Torino) very much. Without them, my training course and my thesis would not have been possible.

Tout d'abord, je voudrais remercier mes parents de m'avoir donné l'opportunité d'aller à l'université. Je les remercie de m'avoir toujours encouragé et de m'avoir supporté lors des moments difficiles. Je leur dédie ce travail.

J'aimerais également remercier le reste de ma famille pour ses encouragements.

Merci aussi à mes amis du Publisquad, que j'ai rencontrés à l'université et qui furent mes compagnons de fortune (la plupart du temps) et parfois aussi d'infortune au cours de mes études.

Je remercie l'université de Namur (FUNDP – University of Namur) et plus particulièrement l'Institut d'Informatique. J'estime en effet avoir reçu une très bonne formation et c'est un bon départ dans la vie. Un tout grand merci aussi au département d'informatique de l'Université de Turin (University of Torino) pour le chaleureux accueil que j'y ai reçu lors de mon stage.

Pour terminer, je veux absolument remercier les professeurs Michaël Petit et Giuseppe Berio, sans qui ce mémoire n'aurait pas pu voir le jour.

Michaël Petit fut mon promoteur dans le cadre de mon stage et de ce mémoire. Il m'a donné l'opportunité de passer plus de trois mois en Italie et a été d'une grande aide autant durant le stage que pendant la rédaction et la correction de ce mémoire. Il a toujours été disponible pour répondre à mes questions et me guider dans mes choix. Un tout grand merci à lui.

Giuseppe Berio fut mon maître de stage. Il m'a accueilli à l'université de Turin et m'a soutenu tout au long du stage. Ses conseils me furent très précieux. J'ai appris beaucoup avec lui et je le remercie de m'avoir consacré autant de temps.

Information about the University of Namur and the University of Turin

If you want to know more about the University of Namur, visit the following website: http://www.fundp.ac.be.

If you want to know more about the Computer Sciences Institute of the University of Namur, visit the following website: http://www.info.fundp.ac.be.

If you want to know more about the University of Turin, visit the following website: http://www.unito.it.

If you want to know more about the Computer Sciences Department of the University of Turin, visit the following website: http://www.di.unito.it.

Table of contents

Table of figures

Table of tables

1. Introduction

1.1 Presentation of the context

Before explaining the goal and the structure of this work, it is necessary to introduce some concepts. The following paragraphs provide a short overview of the context of this thesis.

Modelling consists in creating models. A *model* is a simplified representation (i.e. a more or less formal abstraction) of a *domain* of reality. It allows to reduce the complexity of that domain and in doing so, better understand it. It is also a means of sharing knowledge about the modelled domain. A model is always expressed in terms of a language, known as a *modelling language*. In this thesis, we will use models to represent enterprise or e-business collaborations.

An *enterprise* can be defined in several ways. [Vernadat 1996] gives the following definition: "By nature, enterprises are complex dynamic systems. An enterprise can be perceived as a set of concurrent processes executed on the enterprise means (resources and functional entities) according to the enterprise objectives and subject to business or external constraints". More precisely, we can say that an enterprise is an organization that performs a certain number of activities in order to produce an added value. To accomplish these activities, the enterprise needs a certain amount of inputs (material and human resources). It then performs some operations and produces one or several outputs (products or services). Obviously, these outputs are supposed to bring in a certain profit.

Enterprise modelling is used to study and optimize the structure and processes of enterprises. The content of an enterprise model is whatever the enterprise considers important for its operations. The objective is to support the analysis of the enterprise. As said in [Vernadat 1996], "the main motivations for enterprise modeling are: managing system complexity, better management (control and monitoring) of all types of processes, capitalization of enterprise knowledge and know-how, business process reengineering, and enterprise integration". Nowadays, business process reengineering is often the consequence of the introduction of IT solutions inside an enterprise. Indeed, information systems, by facilitating or automating certain tasks, have a great impact on the enterprise processes. Enterprise modelling greatly helps the integration of such systems in the enterprise.

As mentioned earlier, an enterprise needs inputs and produces outputs. Therefore, it cannot live without collaborating with its environment (customers, suppliers, etc.). These *collaborations* can be accomplished by traditional business or by *e-business* (i.e. electronic business). E-business consists in using information systems and a computer network (Internet for instance) in order to accomplish commercial transactions. As said in [Chappell&al. 2001], "E-business applications are intended to automate some or all of the tasks in the execution of a business process". E-business collaborations are performed by exchanging electronic messages, which contain data and use a certain protocol, between the information systems of the trading partners. Examples of e-business collaborations include the purchase of goods from an electronic catalogue on an Internet site, and the real-time ordering of merchandises between an enterprise and its supplier thanks to a dedicated interface.

E-business modelling is used to describe e-business collaborations. The models provided define the content of the exchanged messages, the order of the messages, the conditions that must be fulfilled for the collaboration to succeed, etc. These models are used as specification for the design of e-business systems. E-business modelling is useful because it allows to standardize electronic collaborations, which is crucial if we want to generalize their use. As [Chappell&al. 2001] says, "If the business interactions that make up a business process are standardized and described in a sufficiently formal way, you can use standard business messages to automate the execution of that business process. This allows companies to do electronic business with all partners that support those e-business processes, rather than just the ones with whom they have existing business agreements".

1.2 Goal of the thesis

Several languages have been proposed for enterprise modelling (examples: IDEF[1], CIMOSA[2], etc.) and e-business modelling (examples: UMM[3], BPSS[4], etc.). Each of these languages provides a set of concepts allowing to define models representing certain specific aspects (for instance, the Entities-Associations language defines the concepts of entity, association, etc., to represent data and information about entities). The concepts proposed by the different languages are often partially redundant but slightly different. Therefore, it is often difficult to choose the most adequate language in a given situation.

Research efforts are currently being done to attempt to define a unique and common language for enterprise modelling (a bit like the UML[5] language, which is intended to be a standard for object-oriented modelling). This language is UEML[6]. UEML stands for "Unified Enterprise Modelling Language" and is the result of the integration of three existing enterprise modelling languages: IEM[6], EEML[6], and Grai[6].

In its current state, UEML allows to model the operational structure of an enterprise but is not able to represent the business collaborations in which the enterprise is involved. This thesis plans to study the feasibility of extending UEML in order to address this lack. UEML would then be able to model both enterprise processes *and* business collaborations, thereby becoming an enterprise *and* e-business modelling language. To reach this goal, it is necessary to identify the relevant (in terms of business collaborations) concepts to be added to UEML. This identification will be performed through a critical study of UMM, which is a good representative of e-business modelling languages.

UMM is the abbreviation of "UN/CEFACT Modeling Methodology" (UN/CEFACT stands for "United Nations Centre for Trade Facilitation and Electronic Business"). This language allows the incremental modelling of business collaborations. The main objectives of UMM are to: facilitate the analysis of e-business processes; help analysts create business collaboration frameworks; understand and formalize the dependencies between processes belonging to business partners in a particular domain; provide reusable process models and descriptions (often common to a whole sector of activity); help the development of quality e-business platforms (ebXML[7] systems for instance); create models that are comprehensible for both domain experts and software developers; create a methodology totally independent of the underlying implementation techniques; and maintain libraries of models.

We can summarize the objective of this thesis in three steps:

- study UEML in details;
- study UMM in details;
- select the concepts to be added to UEML in order to allow e-business modelling with UEML;
- propose an updated version of UEML: EBCML (for "Enterprise and Business Collaboration Modelling Language").

Last remark, it is important to note that the choice of UEML and UMM is not the result of a selection process but is arbitrary. UEML was chosen because the promoters of this work are members of the UEML project. UMM was chosen because it is used in the ebXML framework, which has become a reference in the world of e-business.

---

[1] See http://www.idef.com for more information
[2] See http://cimosa.cnt.pl for more information
[3] See http://www.unece.org/cefact/tmg for more information
[4] See http://www.ebxml.org/specs/ebBPSS.pdf for more information
[5] See [UML site] for more information
[6] See [UEML D3.1 An.] for more information
[7] See http://www.ebxml.org for more information

1.3 Structure of the thesis

This thesis begins with a description of the basic principles needed to enter the world of modelling. We will explain what are modelling languages, models, meta-models and how we can extend a modelling language.

After this introduction, the thesis is divided into three big parts.

The first part focuses on the Unified Enterprise Modelling Language. We will describe the UEML project, the modelling approach, the meta-model, and the modelling constructs. We will also provide a graphical concrete syntax for UEML thereby allowing direct modelling in UEML. Indeed, UEML is currently only an intermediate language used to translate models from one of the integrated languages (IEML, EEML or Grai) to another one. It does not have a concrete syntax of its own and, as such, is not directly useable. We will finish that section by providing a short case study and a review of UEML.

The second part focuses on the UN/CEFACT Modeling Methodology. We will study UMM in details by describing the project, the objectives, the views (with the accompanying meta-models), the modelling constructs, and a summary of the modelling approach. We will finish that section by providing a case study, a review of UMM, and a list of concepts that could be included in UEML in order to model business collaborations with this language.

The third part focuses on EBCML, a new version of UEML allowing e-business modelling. EBCML has been created by extending UEML with collaborative concepts identified during the study of UMM. In that section, we will describe the EBCML meta-model, the modelling concepts, the different views, and the graphical syntax of the language. We will also provide a short case study. The goal of this chapter is to show the feasibility of such an integrated language. It is not meant to be a complete description of the perfect unified language. We will just explain how we can integrate enterprise and e-business concepts into a single modelling language. However, the proposed EBCML will really be useable. Finally, we will provide a conclusion and discuss the possible extensions to this thesis.

1.4 Basic modelling principles

To really understand what we are talking about in this work, it is necessary to explain some basic things about modelling languages, models, meta-models and the extension of a modelling language. These concepts will be used throughout the thesis and their meaning must therefore be clear. The following few paragraphs discuss these topics in the context of UML (Unified Modeling Language). As said in section 1.2, UML is a well-known group of modelling diagrams for object-oriented systems.

1.4.1 Modelling languages

When we want to model a certain concept, we have to choose the modelling language we will use to create the model. A lot of different modelling languages exist but they do not always allow us to represent the same things and they all propose a different kind of representation (which generally consists of graphical objects but not always). The most formal languages that can be used to represent models are mathematics. The less formal (although the richest) languages are natural languages. In between, many forms of languages exist. A modelling language, even if it is graphical, has globally the same properties as a textual language.

A language is defined as a system for communicating. All languages have syntax and semantics.

The syntax is the grammar of the language, a set of rules that must be respected to create a well-formed expression of that language. As explained in [Harel&Rumpe 2000], "we use the term 'syntax' whenever we refer to the notation of the language, including amongst others textual representations and diagrams. Syntactic issues focus purely on the notational aspects of the language, completely disregarding any meaning".

The semantics of a language defines the meaning of the well-formed expressions of the language, in other words the sense of the expressions. As said in [Harel&Rumpe 2000], "the meaning of a language is described by its semantics. It is interesting to note that, in general, computerized tools do not allow us to manipulate semantics directly. Instead, everything we see and work with on the paper or on the screen is a syntactic representation. Expressions are what we use to communicate information. It would be nice if any two communicating participants interpreted expressions of the language in exactly the same way". A semantic definition for a language consists of two parts: a semantic domain and a semantic mapping from the syntax to the semantic domain. Thus a language is composed as described in Figure 1-1:

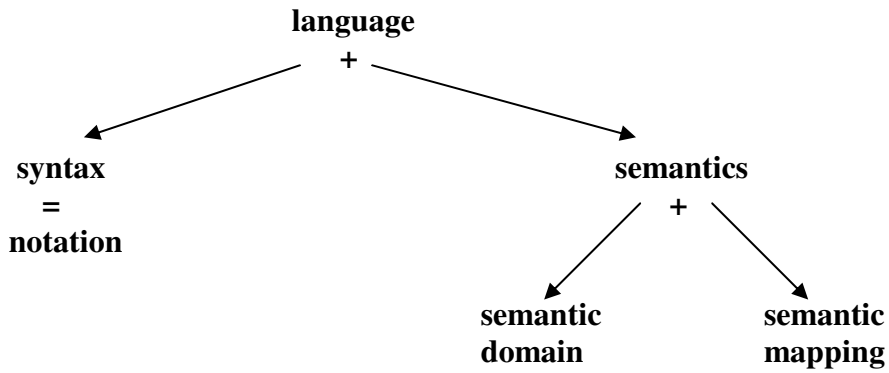


Figure 1 - 1: The structure of a language. [Harel&Rumpe 2000]

The semantic domain consists of the basic elements that exist in our application domain. It describes the important aspects of the systems we want to model. The final step in defining semantics is to relate the syntactic concepts to those of the semantic domain. This is done by defining the semantic mapping. Each syntactic construct is mapped to some semantic element. The semantic interpretation of the syntax is given by this mapping.

Unfortunately, the real border between syntax and semantics is not always very clear. The reason of that problem is that we need a syntactic representation to explain the semantics itself. To properly define a semantic domain we need some kind of language (with its own syntax and semantics) too. The semantics must necessarily always be expressed in a language and therefore be described by the very syntax of that language. So semantics is syntax in some way. We will notice that once more when talking about meta-models in section 1.4.3.

Defining the semantics of a language is something really difficult. For instance, UML 1.4, which has been the reference in object-oriented software modelling for years, had a well-described syntax but did not have a very precise semantics. Its semantics was a bit vague and was not formally defined.

1.4.2 Models

A model is a simplified representation of a part of reality. As said in [Berio&Petit 2003], "modelling always focuses on specific aspects while taking into account all important details about these aspects. Therefore, issued models represent these specific aspects and details while hiding other aspects and details. In other words, modelling and models always refer to some purpose (…). Models can effectively be analysed for understanding the reality, managing complexity and communicating knowledge about some reality".

To create a model, we use a modelling language (which has syntax and semantics as seen before). This language can be purely textual but, for readability reasons, is very often a graphical language (with boxes, arrows, etc.).

Figure 1-2 shows an example of an UML Class Diagram. This model depicts the static structure (the entities and the relationships between them) of a very simple domain.

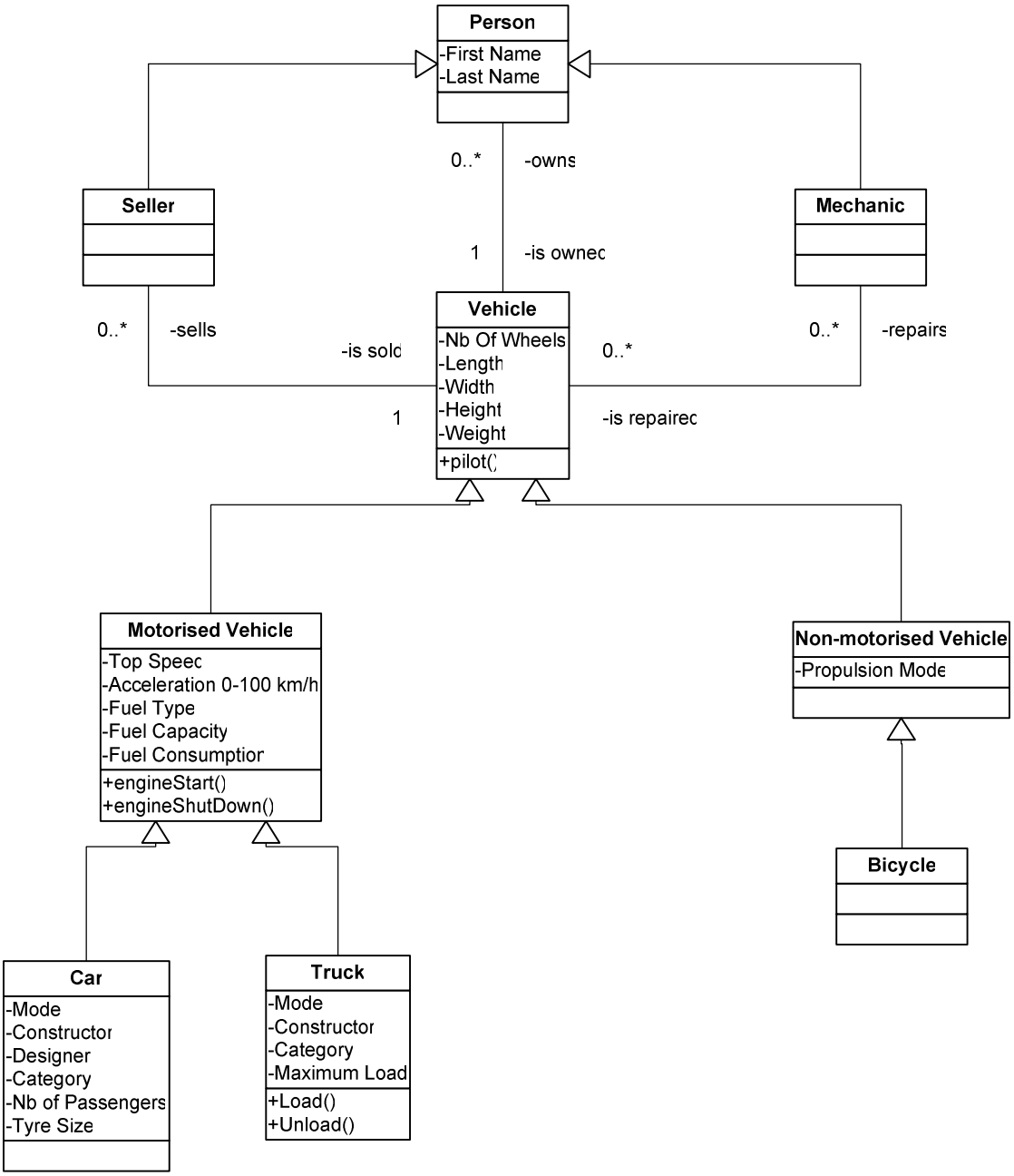


Figure 1 - 2: UML class diagram of a simple domain

This model is a (very) simplified representation of vehicles and what people can do with them. We will not explain the whole model but we can describe a few entities and relationships:

- A Person has the following properties: first name, last name. A Person can own 0 or more Vehicles.
- A Seller is a Person (and has thus a first name, a last name, and can own 0 or more Vehicles). A Seller can sell 0 or more vehicles.
- A Vehicle has the following properties: number of wheels, length, width, height, weight. It is possible to execute the following action on a Vehicle: pilot the vehicle. A Vehicle is owned by only one Person at a time. A Vehicle is sold by only one Seller at a time. A Vehicle can be repaired by 0 or more Mechanics.

UML comprises several diagram types which can represent different aspects of reality. The Class Diagrams show the static structure of the domain. We can add to our model a dynamic view of the engine states. Figure 1-3 shows an UML Statechart that depicts these states and the transitions between them.

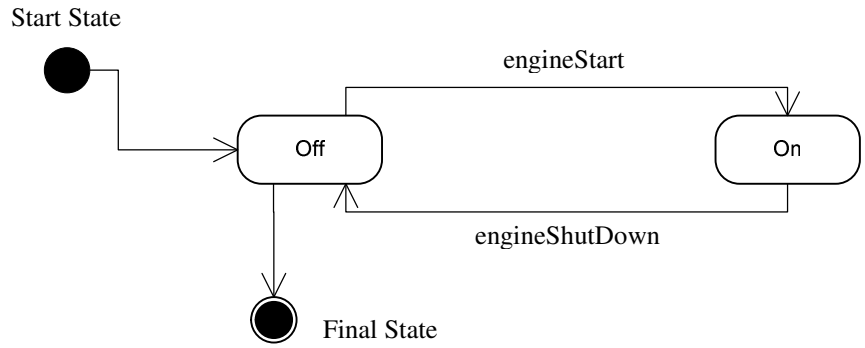


Figure 1 - 3: UML statechart of an engine

This model has the following meaning:

- In the beginning, the engine is in the state "Off".
- If the event "engineStart" happens, the engine goes in state "On".
- If the event "engineShutdown" happens, the engine goes back in state "Off".
- The "Start State" and "Final State" states are compulsory in any UML Statechart but are not really relevant in our discussion.

1.4.3 Meta-models

Actually, it is not easy to give the exact, agreed upon, definition of a "meta-model". [Metamodel site] gives the following two definitions:

- “A metamodel is a precise definition of the constructs and rules needed for creating semantic models”;
- “A metamodel is the collection of ‘concepts’ (things, terms, etc.) that are the vocabulary with which you are talking about a certain domain. Such as Box, Line, Circle, Diagram, or Process, Transition, etc.”.

In our case, we will define a meta-model as a model of a modelling language. In particular, we will use meta-models to describe the syntax or the semantics of a modelling language. The Figure 1-4 hereafter depicts a part of the UML class diagram metamodel. This metamodel provides the rules that must be respected when creating a UML class diagram.

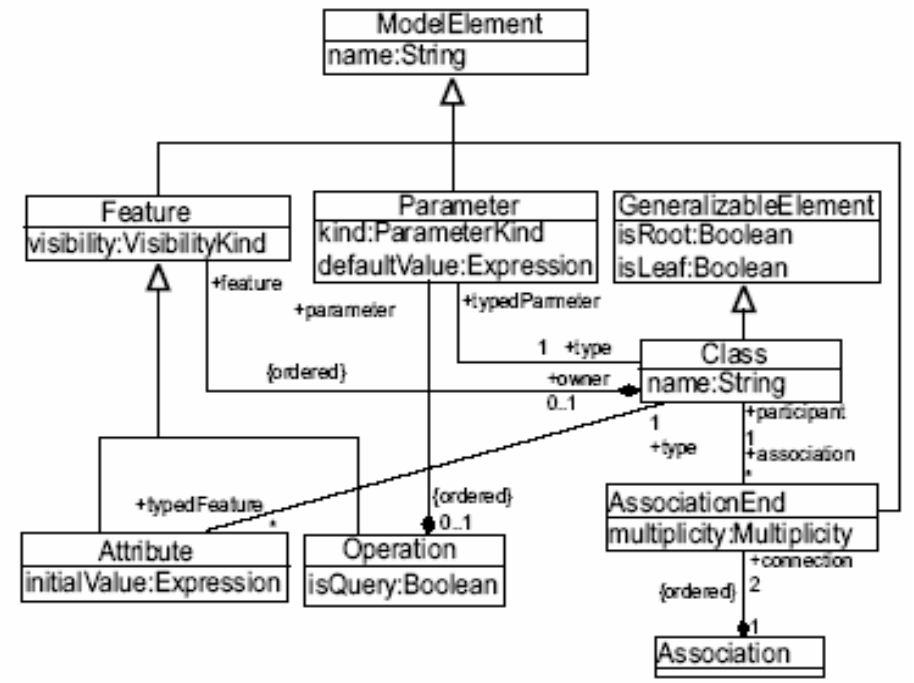


Figure 1 - 4: A part of the UML class diagram meta-model (UML standard v1.4)

The model in Figure 1-2 respects the rules established by the metamodel in Figure 1-4. For example, each class in Figure 1-4 has a name, can have a certain number of attributes and can be involved in a certain number associations.

As seen before, a model is written in a (modelling) language. A meta-model is thus itself written in a language. The modelling language in which the meta-model is defined may even be a subset of the modelling language that the meta-model must describe. This may be difficult to conceive but it is quite usual. For example, English is itself defined in English: the grammar rules (the correct structure of a sentence for instance) are described in grammar books written in English, and each English word has an English definition in an English dictionary. Here we can really see the difficulty to put a border between syntax and semantics: the modelling language used by a meta-model to define a modelling language may itself be defined in a meta-meta-model. This meta-meta-model has to be described in a language too. For example, we can describe English grammar and vocabulary in French, but to understand that description, we have to describe French too. Semantics can never be fully described since it always relies on the use of another language and the semantics of that other language can neither be fully described.

1.4.4 Extending a language

It is also important to notice that a language needs an extension system. It must always be possible to introduce new concepts in the language. For instance, English and French are living languages which means they are always evolving. Introduction of new words and even grammar changes can happen. A system must exist to allow these changes. As said in [Harel&Rumpe 2000], a language extension must always be explained in terms of the language itself. This allows a communication partner to understand the new concept. Thus, an extension gets its semantics indirectly through the semantics of the explanation. One common mechanism of extension relies on binding a subexpression of the language to a name, where the subexpression has already been given semantics. For example, we will study the UMM modelling language in section 3. This language is described by a meta-model written in UML. The UMM meta-model is not only written in UML, it intends to extend UML by using UML stereotypes. UML stereotypes constitute the extension system of UML. A stereotype creates a new modelling element from an existing one by giving it an extended, user-defined, meaning.

We can find the following definition of "stereotype" in the UML v1.5 specification : "A stereotype is, in effect, a new class of metamodel element that is introduced at modelling time. It represents a subclass of an existing metamodel element with the same form (attributes and relationships) but with a different intent. Generally a stereotype represents a usage distinction. A stereotyped element may have additional constraints on it from the base metamodel class. It may also have required tagged values that add information needed by elements with the stereotype. It is expected that code generators and other tools will treat stereotyped elements specially. Stereotypes represent one of the built-in extensibility mechanisms of UML". We can see that UML allows us to specialize the meaning of certain elements through stereotypes but that it does not describe the meaning of the new stereotypes within the language itself. Instead, informal English definitions are often used or even definitions that appeal to specific tools. The UML v1.5 extension system is therefore not formally defined.

UML stereotypes can be used to add a new modelling element to the example shown in figure 1-2. We can add a relationship "More used than" to the basic UML. When this relationship is used, it means that a specific vehicle (an instance of Vehicle) is more used than some other specific vehicle. The model in Figure 1-5 uses this new relationship.

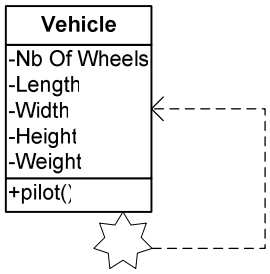


Figure 1 - 5: Introduction of a new relationship "More used than"

This model will only be acceptable if we update the metamodel of our modelling language. This can be done by adding a new UML stereotype to the metamodel. Figure 1-6 shows how the UML metamodel achieves this.

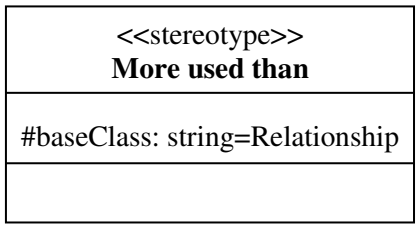


Figure 1 - 6: The "More used than" stereotype

By adding this stereotype to our UML metamodel, we have extended UML.

2. UEML

In this section, we will study the Unified Enterprise Modelling Language (UEML). We will first provide a short overview of UEML, then we will describe the strategy used to create it, its meta-model, and its most important concepts. Afterwards, we will present a proposal of graphical syntax for UEML and present a short case study. Finally, we will provide a review of UEML.

2.1 Overview of UEML

The idea of creating UEML had been discussed in several ICEIMT[1] conferences in 1992 and 1997. The objective of these conferences was to improve enterprise integration through better enterprise modelling. Integration needs interoperability, which is really a critical element to increase the effectiveness of enterprise collaboration and flexibility. But until then, the focus had been set on software interoperability (through XML solutions, etc.) which concentrates on format compatibility. This kind of interoperability has the major drawback of ignoring the modelling domain. The modelling domain is the environment that needs the help of software, the real-life context that provides the reasons why the software is built. Real interoperability cannot be reached without proper understanding of the modelling domain. This larger kind of interoperability does not take into account any question of technical implementation. Software interoperability is based on modelling domain interoperability and not the opposite. Enterprise interoperability cannot be achieved without understanding the enterprise domain. UEML aims at helping this understanding by providing means of representing the enterprise domain.

The first version of UEML was created within the UEML Project. As stated in [Berio UEML D3.1], the primary goal of the UEML project was to define a modelling approach allowing a greater enterprise integration, flexibility and interoperability (by reducing the effort needed to reach enterprise collaboration). However, the objective of the UEML project was not to create a final product (i.e. a really usable integrated modelling language). Its goal was rather to:

- study the feasibility of such an integrated language;
- define a first version of this language (UEML 1.0) to be later improved.

[Berio UEML D3.1] says: "some of the main objectives, stated in the *first phase* of this project, are that the UEML should (i) *capitalise the existing knowledge about enterprise modelling* and (ii) *make existing modelling tools more interoperable* by enabling some kind of exchange between these tools (though, the UEML will probably not be able to solve alone all the problems related to this *kind of interoperability*). It should be noted that these two points are complementary and both are needed".

More concretely, the UEML project wanted to (amongst other objectives):

- integrate several existing enterprise modelling languages in a unified one (each of the modelling languages to be integrated was implemented in a dedicated modelling tool);
- define mappings between the original languages and the unified one. These mappings were used as specifications for mechanisms allowing to export models from one tool (using one of the original languages) to another (using another language).

The UEML 1.0 produced by the UEML project has some limitations. For instance, it is currently only a model translation[2] medium and does not have a concrete syntax of its own. It is therefore not possible to directly model an enterprise with UEML. If it is meant to be used as a real

[1] International Conference of Enterprise Integration and Modelling Techniques
[2] Between IEM, EEML, and Grai, which are the languages that UEML integrates

modelling language, UEML should possess a syntax and a semantics. UEML 1.0 has an associated meta-model that describes its abstract syntax. However, it does not have a formal semantics[1]. The usefulness and even the feasibility of such a formal semantics are not certain. This would be very complex and would require that a consensus is reached about the syntax and the semantics of UEML.

2.2 The strategy for creating UEML

The integration of concepts from different enterprise modelling languages is not a simple process. In the UEML project, a scenario-based methodology was defined in order to allow the incremental creation of the UEML. It can be summarized as consisting in two big tasks:

1. select a case study (called "scenario") and to model it in the modelling languages to be integrated. These models were made by experts in the studied languages.
2. compare the resultant models to see what was common among the languages to be integrated. The three languages were afterwards meta-modelled in UML and the process of unification began.

For the UEML v1.0, three enterprise modelling languages were chosen for the integration. These languages are IEM, Grai and EEML[2]. The "scenario" was first modelled in IEM and was then remodelled in Grai and EEML. The Figure 2-1 shows the complete UEML strategy.

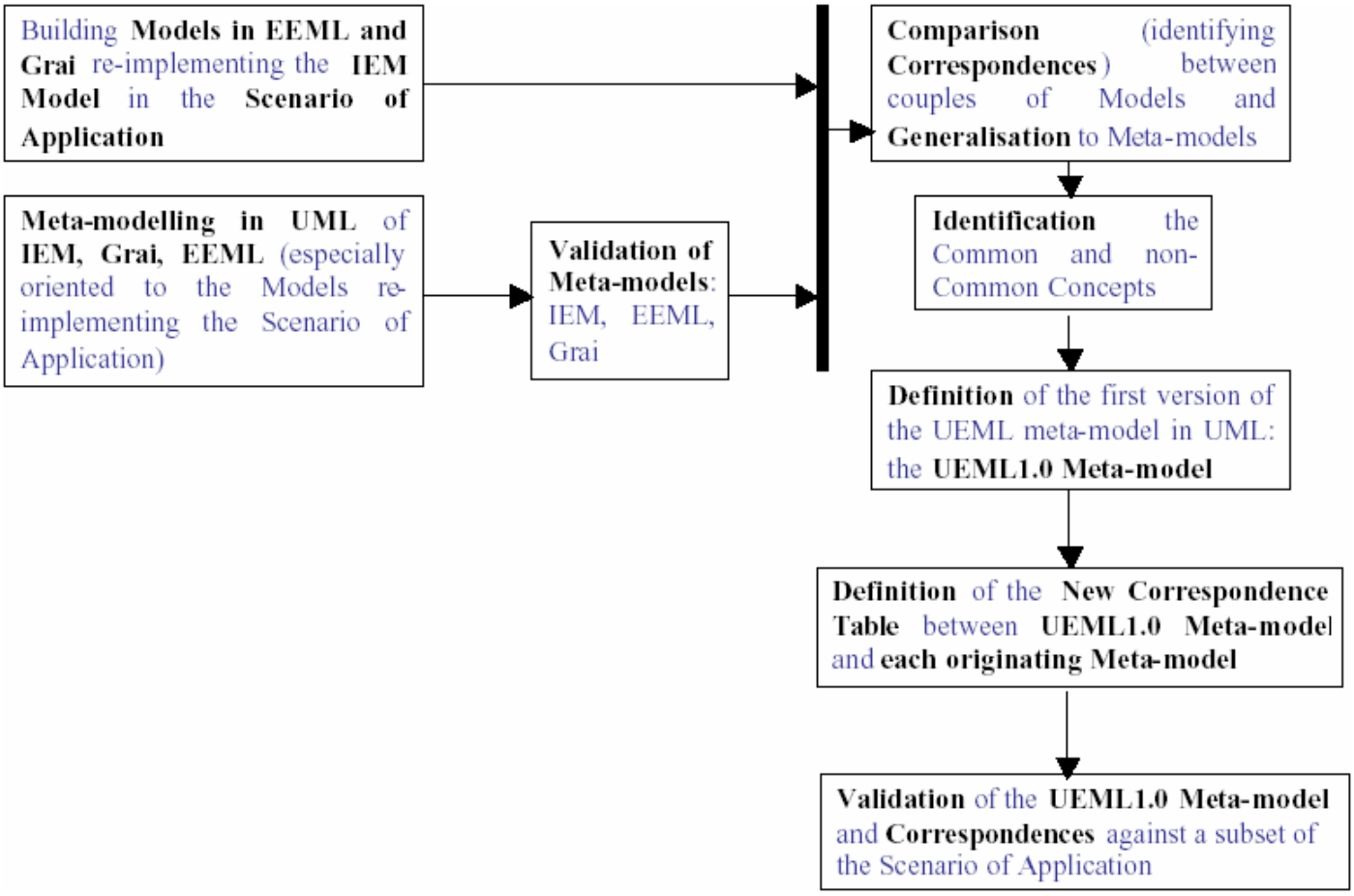


Figure 2 - 1: The strategy for UEML (from [UEML D3.1 An.])

[1] It is important to note that UEML is not an exception in this field. Many other modelling languages only possess an informal (i.e. in common language) semantics.

[2] IEM is the language supported by the modelling tool Moogo (designed by IPK); EEML is a language supported by the modelling tool Metis (designed by Computas); GRAI is the language supported by the modelling tool eMagim (designed by Graisoft).

The first execution of this strategy has provided the UEML 1.0 which contains the core constructs of the unified enterprise modelling language. These core constructs were necessary to allow the exchange of models between the three studied modelling tools (Moogo, Metis and eMagim). In the future, the UEML could be extended by applying the same strategy with two new inputs: a new enterprise modelling language and the UEML 1.0 itself.

Meta-modelling has been used to define the capabilities of the integrated languages as well as those of the UEML, therefore allowing the creation of mappings between all these languages. The language chosen for meta-modelling was UML. UML was chosen because it was the most known modelling language and it was particularly adapted to represent the abstract syntax of the studied languages.

[UEML D3.1 An.] explains a method for finding correspondences between meta-models of two different modelling languages. This method is based on the study of a scenario (a unique case study modelled with each of the compared modelling languages).

The definition of the UEML meta-model was based on the research of common and non-common concepts in the meta-models of the integrated languages. Synthetically put, two concepts belonging to two different languages can be considered as common if and only if, in the scenario models, they are always used to model the same phenomena of the real world. It is obvious that the common concepts had to be present in UEML. Indeed, since these concepts belong to all the studied languages, there is strong evidence that these concepts are relevant in the domain of enterprise modelling. However, some useful non-common concepts have been integrated as well.

The commonality of concepts is in itself difficult to evaluate because it refers to a problem of semantics. If we want to be general, we can say that concepts from two different modelling languages (each with its own semantics) are common if their meaning is the same. But it is not that easy in practice. What does "commonality" concretely mean? Equivalence? Inclusion? Intersection? At first glance, there are several possible definitions and the meaning of “commonality of concepts” should be a subject of research.

The approach taken in the UEML strategy was to find semantic correspondences (equivalence, inclusion, etc.) between syntactical elements (such as “activity”, “flow”, etc.) from the studied modelling languages. In order to find these correspondences, and since there is no formal semantics for the studied languages, it was necessary to work with a scenario.

To create the UEML meta-model, the concepts from the integrated languages have been classified as common or non common in the following way:

- *Equivalent Concepts* between all the meta-models were made *Common Concepts*
- *Equivalent Concepts* between one or two meta-models only were made *Non Common Concepts* but were considered in the UEML meta-model**.**
- Concepts which were not related were made Non Common Concepts
- Concepts that were not relevant were not retained.
- *Overlapping Concepts* needed further study. Overlapping concepts are those that are sometimes used to represent similar real world phenomena, but that are also sometimes used to represent different real world phenomena. *Overlapping Concepts* required further treatment to be reduced to *Equivalent Concepts* by constraining (i.e. specialising) their use.

2.3 UEML meta-model

Figure 2-2 shows the UEML 1.0 meta-model. The designers of this meta-model have chosen, for readability reasons, to show the inherited attributes in the subclasses.

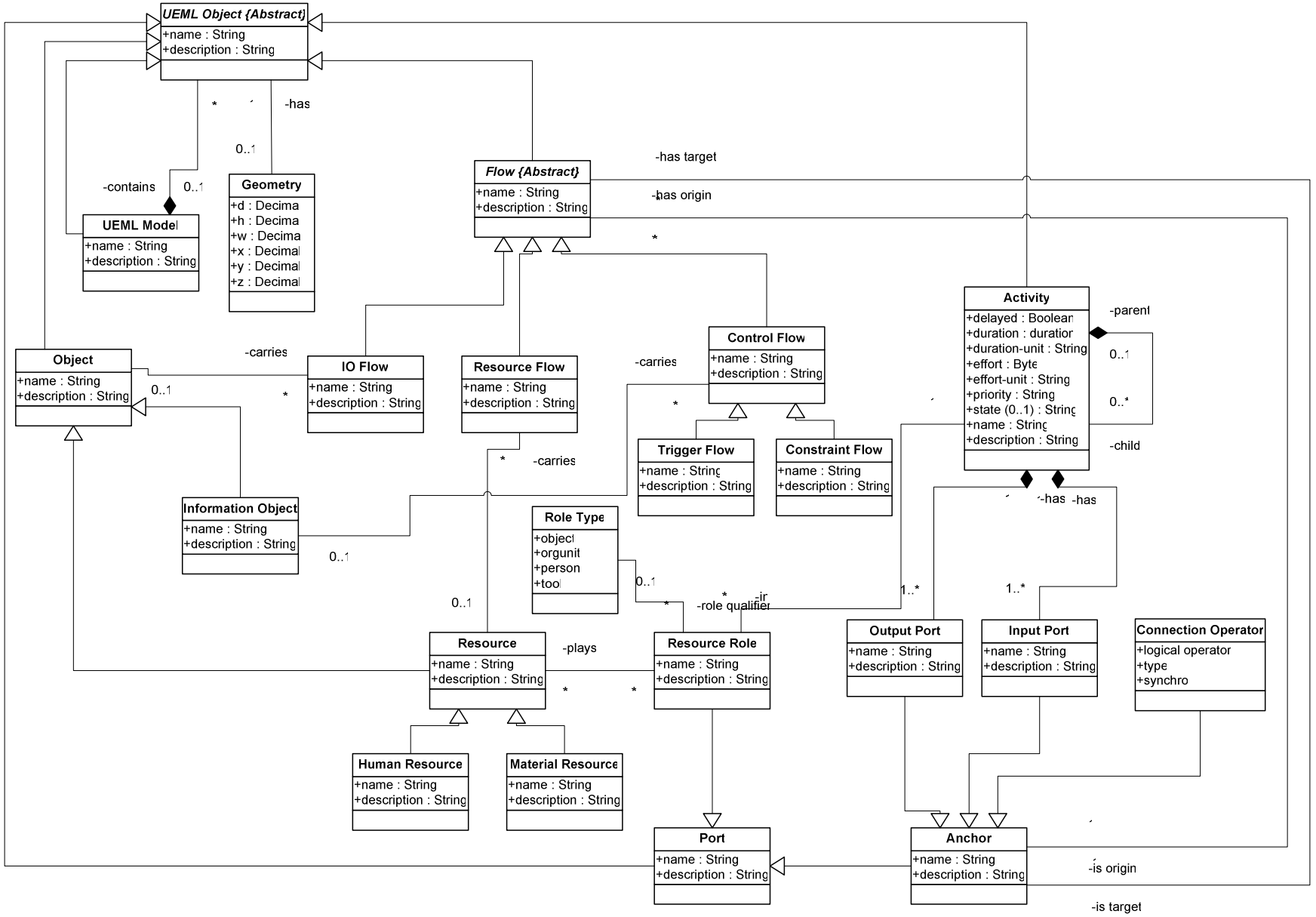


Figure 2-2: UEML meta-model

2.4 Description of the main UEML concepts.

Here follows a short overview of the major UEML 1.0 concepts. The complete descriptions can be found in [UEML D3.1 An.].

Activity

An *Activity* represents a generic description of a part of the enterprise behaviour that produces outputs from a set of inputs. An *Activity* may be controlled and may use resources. It represents a set of similar activities executions.

- An *Activity* may be decomposed into *Activity(ies)*.
- An *Activity* may require one or several *ResourceRole(s)* played by *Resource(s)*. A *Resource* may be required by several *Activities* (and thereby play several *ResourceRoles*). A *ResourceRole* is specific to an *Activity*.
- An *Activity* is at least related to one *InputPort* to which flows representing the inputs of the activity are connected.
- An *Activity* is at least related to one *OutputPort* to which flows representing the outputs of the activity are connected.
- An *Activity* cannot be composed of itself (directly or indirectly).
- There are two ways to represent the fact that a *Resource* is used in an *Activity*. Either through the definition of a *ResourceRole* that a *Resource* plays in the *Activity*, or through a *Flow*, connected to the *InputPort* of the *Activity*, which carries the *Resource*. The first possibility should be used when the origin of the *Resource* is not explicitly represented (if the resource is not the subject of an explicit flow). The second should be used when the origin of the Resource is explicit (e.g. a flow coming from another *Activity*).

Flow, IOFlow, ResourceFlow, ControlFlow, TriggerFlow, ConstraintFlow

A *Flow* represents the flowing of an *Object* from an origin to a target. The origin and the target of a flow are *Anchor(s)* that can be *InputPort(s)*, *OutputPort(s)* or *ConnectionOperator(s)*.

A *Flow* is either an *IOFlow*, a *ResourceFlow* or a *ControlFlow*. A Flow carries *Object(s)* that are compatible with its most specialized type (an *IOFlow* carries *Object(s)* or *InformationObject(s)* or *Resource(s)*, a ResourceFlow carries *Resource(s)***,** a *ControlFlow* carries *InformationObject(s)*).

An *IOFlow* represents the flowing of an *Object* (a product for instance) between two *Activity(ies)*, the *Object* being an input and/or output for the *Activity(ies)***.** The Object can possibly be consumed, modified or created by the *Activity*. If the *IOFlow* is connected to an *InputPort* of an *Activity*, the Object that is carried by this *Flow* is supplied to the *Activity* to be executed. If the *IOFlow* goes out (either directly or indirectly) from an OutputPort *of* an *Activity***,** the *Object* carried by this *Flow* is produced by the *Activity*. A *ResourceFlow* mainly represents the flowing of a *Resource* between two *Activity(ies)*. The *ResourceFlow* then connects an *OutputPort* of an *Activity* that produces it to an *InputPort* of the *Activity* that requires it. A *ControlFlow* connecting two *Activity(ies)* can represent three different things:

(1) a precedence relationship between *Activity(ies)* (a *ControlFlow* that does not carry any Object);

(2) the triggering of an *Activity* after another (a *TriggerFlow*, possibly carrying an *InformationObject* that represents the event that triggers the target *Activity*);

(3) the flowing of an *InformationObject* that is used to constrain the execution of the *Activity* (a *ConstraintFlow* carrying a constraining *InformationObject* such as, for instance, a description of a procedure to be followed to execute the *Activity*).

- An *IOFlow* carries at most one *Object*. However, an *Object* can be carried by several *IOFlow(s).*

- A *ResourceFlow* carries at most one *Resource*. However, a *Resource* can be
- carried by several *ResourceFlow(s)*.
- A *ControlFlow* carries at most one *InformationObject*. However, an *InformationObject* can be carried by several *ControlFlow(s)*.
- A *Flow* cannot have the same *Anchor* as origin and destination.
- A ConstraintFlow carries at least one InformationObject.
- *Flow(s)* connected to the same *ConnectionOperator* have to belong to the same subclass of the *Flow* class.

Anchor
An *Anchor* is the origin or the target of a *Flow*. It is an *OutputPort*, an *InputPort* or a *ConnectionOperator*. *Flow(s)* can only be attached to *Anchor(s)***.** This implies that for defining *Flow(s)* coming from or going to an *Activity*, an *InputPort* or *OutputPort* has to be defined and the *Flow* has to be attached to this port.

InputPort
An *InputPort* represents the entrance of a *Flow* in an *Activity*. It is a special kind of *Anchor*. An *InputPort* is a component of exactly one *Activity*.

OutputPort
An *OutputPort* represents the exit used by a *flow* to leave an *Activity*. It is a special kind of *Anchor*. An *OutputPort* is a component of exactly one *Activity*.

ConnectionOperator
A *ConnectionOperator* represents the grouping or splitting of *Flow(s)* between two or more *Activity(ies)***.** It is a special kind of *Anchor*. It can be of several types:

- A *ConnectionOperator* of type « Join » is the target of at least two *Flow(s)* and is the origin of exactly one *Flow*.
- A *ConnectionOperator* of type « Split » is the origin of at least two *Flow(s)* and is the target of exactly one *Flow*.

The joining/splitting of flows can be used to represent logical operations such as AND, OR, and XOR.

ResourceRole
A *ResourceRole* defines the need for *Resource(s)* in an *Activity*, that is, it defines the role played by *Resource(s)* in an *Activity*.

- A *ResourceRole* is used in exactly one *Activity*. An *Activity* may use any number of *ResourceRole(s)*.
- Several *Resource(s)* can play a *ResourceRole*. A *Resource* can play several *ResourceRole(s)*.
- A *ResourceRole* can be defined independently of any *Resource*.

Resource, MaterialResource, HumanResource
A *Resource* is a special kind of *Object* needed for the execution of an *Activity*. It can represent either a single *Resource* or a set of *Resource(s)* with similar capabilities and properties. A *Resource* may be a *MaterialResource* or a *HumanResource*.

- A *Resource* may play (any number of) *ResourceRole(s)*.
- A *Resource* can be involved in (any number of) *ResourceFlow(s)*.

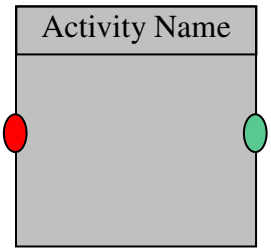
Object, InformationObject
An *Object* is something that can be attached to a *Flow*. In other words, it is something that may be needed or produced by an *Activity*. It can be an *InformationObject* or a *Resource*.
An *InformationObject* is a kind of *Object* made only of information.

- An *IOFlow* carries at most one *Object*. However, an *Object* can be carried by several *IOFlow(s)*.
- A *ControlFlow* carries at most one *InformationObject*. However, an *InformationObject* can be carried by several *ControlFlow(s)*.

2.5 A graphical concrete syntax for UEML

As mentioned before, UEML is currently only a model exchange language. In the UEML project, only its meta-model and a corresponding XML syntax were defined. It is therefore not possible to really model enterprises in UEML. UEML deserves more than this status and should be a "real" useable enterprise modelling language. A concrete syntax is thus needed.

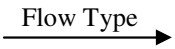
In what follows, we have defined a graphical concrete syntax for UEML. The use of this syntax will be illustrated in a short case study further in this chapter.



An *Activity*

An *Input Port*

An *Output Port*



A *Flow*. It can be *IO Flow*, Re flow (*Resource Flow*), Cl flow (*Control Flow*), Tr Flow (*Trigger Flow*), Ct Flow (*Constraint Flow*)

An "AND" *Connection Operator*

An "OR" *Connection Operator*

A "XOR" *Connection Operator*

A *Human resource*. "Worker" is only an example of human resource.

A *Material Resource*. "Product" is only an example of material resource.

An *Information Object*

A "person" *Resource Role Type* (an employee of the modelled enterprise). {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}


orgunit

An "orgunit" *Resource Role Type* (a group of people from our enterprise or another company with which we collaborate but do not make e-business). {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}


tool

A "tool" *Resource Role Type*. {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}

This syntax is sufficient to model in UEML but it would be uncomfortable to make all the drawings by hand. The syntax could be more useful if it were implemented in a software modelling tool. Creating a dedicated tool would take much time and was not in the scope of this thesis. So, a prototype of such a tool was created by integrating the UEML modelling elements into an existing tool: Microsoft Visio. A Microsoft Visio stencil, which is a toolbar, was created and dedicated to UEML.

The user selects the modelling element he/she wants to add to the model and then, he/she uses the drag and drop interface to put the item at the appropriate place on the sheet. The user is then able to customize each item in order to describe the domain he/she is modelling. The following figure shows a screenshot of a modelling session with UEML in Microsoft Visio:

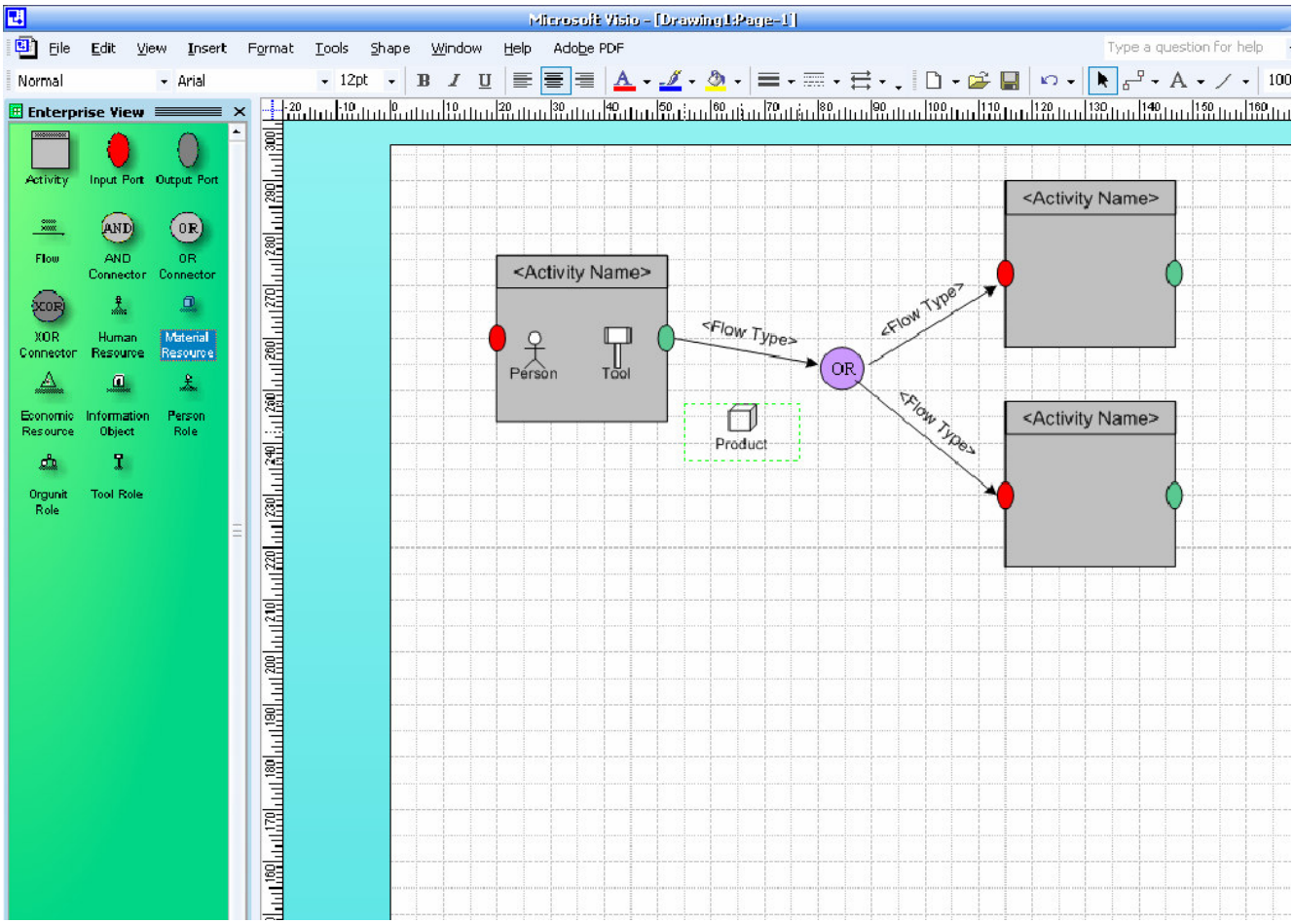


Figure 2 - 3: Modelling in UEML with Microsoft Visio

2.6 Case study

We will now introduce a simple example showing the use of UEML with its graphical concrete syntax (which was described earlier in this chapter). We will model a very simplified vision of the operational structure of a temping agency.

A temping agency is providing temporary workers to its customers, which are enterprises that need human resources. The customer asks the temping agency for defined profiles and this last finds the best suited applicants for the job. The temporary workers are paid by the temping agency. It is itself paid by the customer that uses the workers.

We can describe the business of the temping agency by dividing it into five major *activities*. The following figure shows the workflows between these main *activities*:

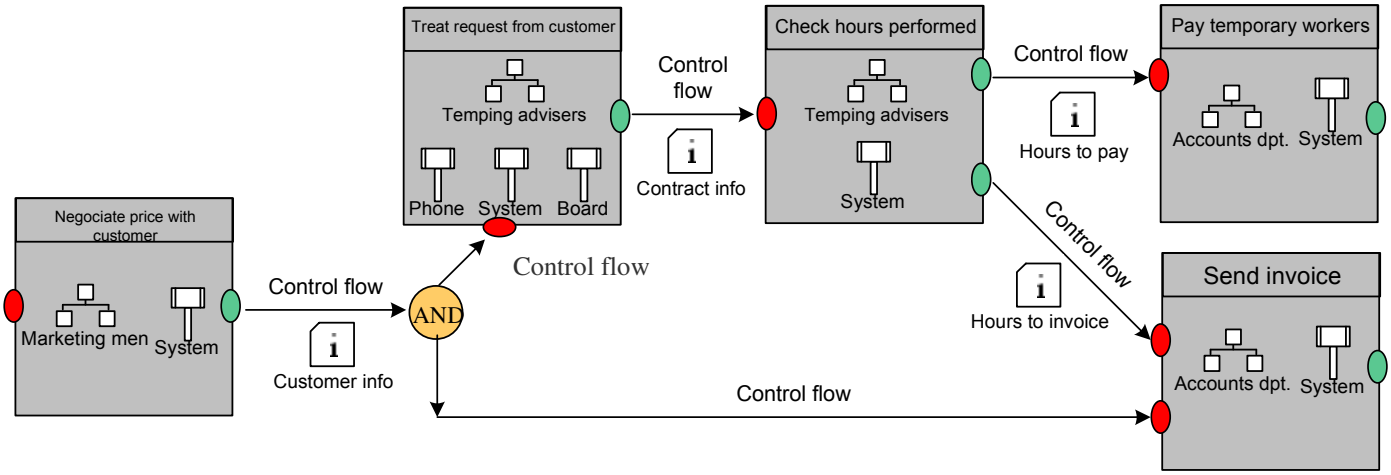


Figure 2 - 4: *Flows* and *activities*[1]

Here follows a brief description of each *activity*:

- "Negociate price with customer" consists in negotiating with the customer to define the price that will be charged. This price is determined according to several factors. This *activity* is performed by marketing men. An information system is used during this *activity*. The output of this *activity* is an information object containing pieces of information about the customer and the contract to be established.
- "Treat request from customer" consists in satisfying the customer's needs for *human resources*. This *activity* is performed by temping advisers. The phone, a board and an information system are used during this *activity*. The input of this *activity* is an *information object* containing pieces of information about the customer and the customer's contract. The output of this *activity* is an *information object* containing pieces of information about the contract established with the temporary worker.
- "Check hours performed" consists in checking the number of hours that the worker has done during the week. This *activity* is performed by temping advisers. An information system is used during this *activity*. The input of this *activity* is an *information object* containing pieces of information about the contract established with the temporary worker. It has two outputs: an *information object* telling the number of hours the worker has to be paid for, and an *information object* telling the number of hours the agency can charge its customer.

[1] An "AND" connector could be used at the entrance of the "Send Invoice" activity instead of using two input ports. However, the meta-model allows both possibilities and we wanted to illustrate this fact.

- "Pay temporary workers" consists in paying the temporary worker for the number of hours that he/she has worked. This *activity* is performed by the accounts department of the temping agency. An information system is used during this *activity*. The input of this *activity* is an *information object* telling the number of hours that the worker has to be paid for.
- "Send invoice" consists in charging the customer for the number of hours that the temporary worker has worked. This *activity* is performed by the accounts department. An information system is used during this *activity*. The input of this *activity* is an *information object* telling the number of hours that the customer has to be charged for.

Each of these *activities* can itself be divided into subactivities linked by *flows*. The following figure shows the decomposition of the "Treat request from customer" *activity*.

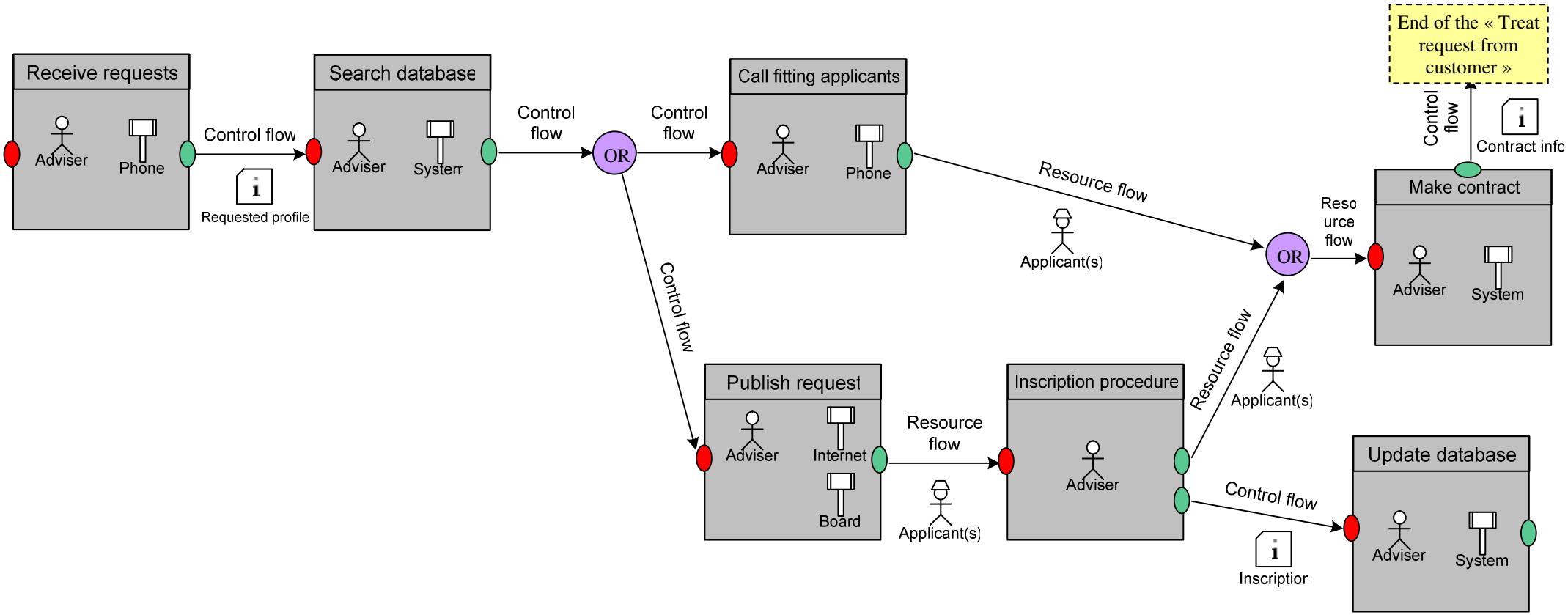


Figure 2 - 5: Decomposition of the "Treat request from customer" activity[1]

[1] "End of the « Treat request from customer » activity" is not part of the model. It only indicates that the model shown here is incomplete because it only models the "Treat request from customer" activity.

Here is a short description of each of these subactivities:

- "Receive request" consists in answering the phone calls coming from the customers and in listening to their needs. This *activity* is performed by temping advisers. The phone is used during this *activity* (obviously, other means of communication, such as fax or e-mail, could be used but, for simplicity reasons, we have chosen not to represent them). The output of this *activity* is an *information object* containing pieces of information about the requested profile.
- "Search database" consists in looking for the requested profile in the list of registered applicants. This *activity* is performed by temping advisers. An information system is used during this *activity*. The input of this *activity* is an *information object* containing pieces of information about the requested profile. This *activity* has no output. A Control Flow is used to show that this *activity* must be performed before the "Call fitting applicants" and "Publish request" *activities*.
- "Call fitting applicants" consists in calling the best suited applicants found in the database. This *activity* is performed by temping advisers. The phone is used during this *activity*. The output of this *activity* is a *human resource*: the applicant(s).
- "Publish request" consists in publishing the requested profile in the hope of finding new applicants. This *activity* is performed by temping advisers. Boards (to stick up posters) and Internet are used during this *activity*. The output of this *activity* is a *human resource*: the applicant(s).
- "Inscription procedure" consists in helping the new applicants to fill in the inscription form. This *activity* is performed by temping advisers. The input of this *activity* is a *human resource*: the applicant(s). This *activity* has two different outputs: *human resource* (the applicant(s)) and an *information object* containing pieces of information coming from the inscription form.
- "Update database" consists in updating the list of applicants by adding a new applicant in the database. This *activity* is performed by temping advisers. An information system is used during this *activity*. The input of this activity is an information object containing pieces of information coming from the inscription form.
- "Make contract" consists in establishing a contract with the best suited applicants for the requested profile. This *activity* is performed by temping advisers. An information system is used during this *activity*. The input of this *activity* is a *human resource*: the applicant(s). The output of this *activity* is an *information object* containing pieces of information about the contract established with the applicant(s).

This case study is of course very simple and does not reflect all the possibilities offered by UEML for modelling enterprises. As said before, UEML is the result of the comparison of three enterprise modelling languages and is therefore similar to the well-known languages in that domain. However, this little case is enough to introduce the use of the proposed graphical concrete syntax for UEML.

Chapter 4 will introduce an attempt at combining some aspects from both UMM and UEML into a single modelling language. The current version of UEML, UEML 1.0, will be totally reused in that new modelling language. You will thus have the opportunity to see other examples (using other modelling elements) showing the use of UEML and of its graphical syntax.

2.7 Review of UEML

2.7.1. UEML problematic issues

The version of UEML we have studied, UEML 1.0, is not a final modelling language. For this reason, there are some aspects of enterprises that cannot be modelled in UEML. The most interesting example is that it is currently not possible to represent the e-business collaborations in which the modelled enterprise is involved. Even if UEML 1.0 could be used for modelling processes spanning

over several organisations, it lacks constructs (concepts such as event, interaction, etc.) to model trading collaborations. The goal of this thesis is precisely to solve this problem.

UEML 1.0 does not have a concrete syntax and is therefore not directly useable. Until now, UEML has only been used as a model exchange medium between the integrated languages. It was its primary objective but it is obvious that UEML could be used as an enterprise modelling language in its own right. This is why an adapted graphical syntax has been provided in this chapter.

Some multiplicities in the UEML meta-model are not constraining enough. This is because UEML is used to exchange models from different languages. It must not be too constraining. But it sometimes allows strange things while directly modelling in UEML. For instance, the UEML meta-model allows to use either a connection operator to join flows at the entrance/exit of an activity or use several input/output ports (one for each of the coming flows).

2.7.2. UEML strengths

UEML gathers together the common modelling concepts coming from three different enterprise modelling languages. It therefore represents a sort of consensus in enterprise modelling. For this reason, UEML is particularly well suited to model the functioning of enterprises.

The “strategy for UEML” can be applied to add new concepts (coming from other enterprise modelling languages) to the UEML 1.0 thereby increasing the modelling range of UEML. This strategy, presented in section 2.2, has been successfully applied to three modelling languages and is therefore effective. However, we should be careful and not apply the “strategy for UEML” with modelling languages that are too specific. This would increase the amount of “non-common concepts” retained and make UEML more complicated. The mappings between the integrated languages would also become very complex.

The scenario-based approach seems to be effective to compare modelling languages. It is practical and allows quick comparison of concepts in a given context. Because of the absence of formal semantics in the integrated languages, it was difficult to proceed differently for UEML 1.0. Of course, the scenario must be carefully chosen and be representative enough of the languages abilities.

The official documentation introducing UEML is well made and explains well the major concepts and the methodology used to create the UEML 1.0. It is therefore not difficult to enter the UEML world.

2.8 Final word about UEML

UEML results from the integration of three enterprise modelling languages. It has been built by using a strategy based on the identification of common concepts in these three languages. Some useful non-common concepts have been included as well. Therefore, it is obvious that UEML is a relevant language for enterprise modelling. However, UEML does not allow to model (e-) business collaborations. In chapter 3, we will study UMM, an e-business modelling language, in order to identify the collaborative concepts that could be added to UEML. In chapter 4, we will propose a new version of UEML, called EBCML, allowing both enterprise modelling and (e-) business collaborations modelling.

3. UMM

In this section, we will study the UN/CEFACT[1] Modeling Methodology (UMM), which is used to model business collaborations. We will first provide a short overview of UMM, then we will describe its objectives, its modelling views, and its modelling approach. Afterwards, we will present a short case study and a review of UMM. Finally, we will select the UMM collaborative concepts that could be added to UEML.

3.1 Overview of UMM

UMM allows the incremental modelling of business collaborations. Business collaborations are business processes involving information exchanges, in a technology-independent way, between trading partners. UMM uses a subset of the phases and workflows proposed in the Rational Unified Process (RUP). RUP describes the steps in which all software development projects pass through. It is made of four major phases: inception, elaboration, construction, and transition. Between the initial idea and the delivery of the operational information system, the project is further divided into a series of workflows.

UMM is limited to the inception (documentation of the idea underlying the project) and elaboration (refinement and expansion of the idea) phases. Indeed, UMM is only useful in the business modeling, requirements, analysis, and design workflows. Figure 3-1 shows where UMM is used in the RUP framework.

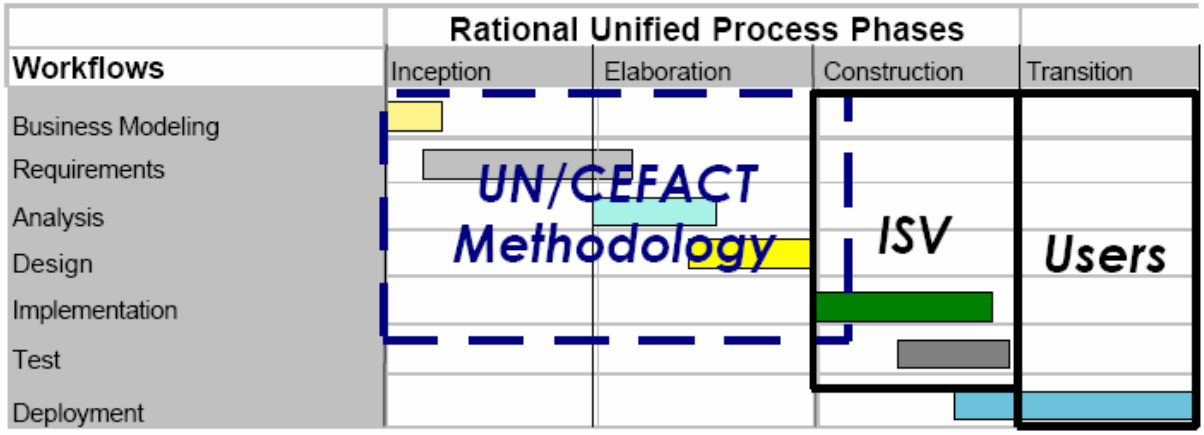


Figure 3 - 1: Place of UMM in the RUP framework (from [UMM-N090R10])

Furthermore, UMM is used in the open-edi framework. [UMM-N090R10] defines open-edi as the "electronic data interchange among multiple autonomous organizations to accomplish an explicit shared business goal according to Open-edi standards (i.e. that complies with the Open-edi Reference Model Standard - ISO/IEC 14662)". The open-edi reference model is depicted in figure 3-2.

[1] United Nations Centre for Trade Facilitation and Electronic Business

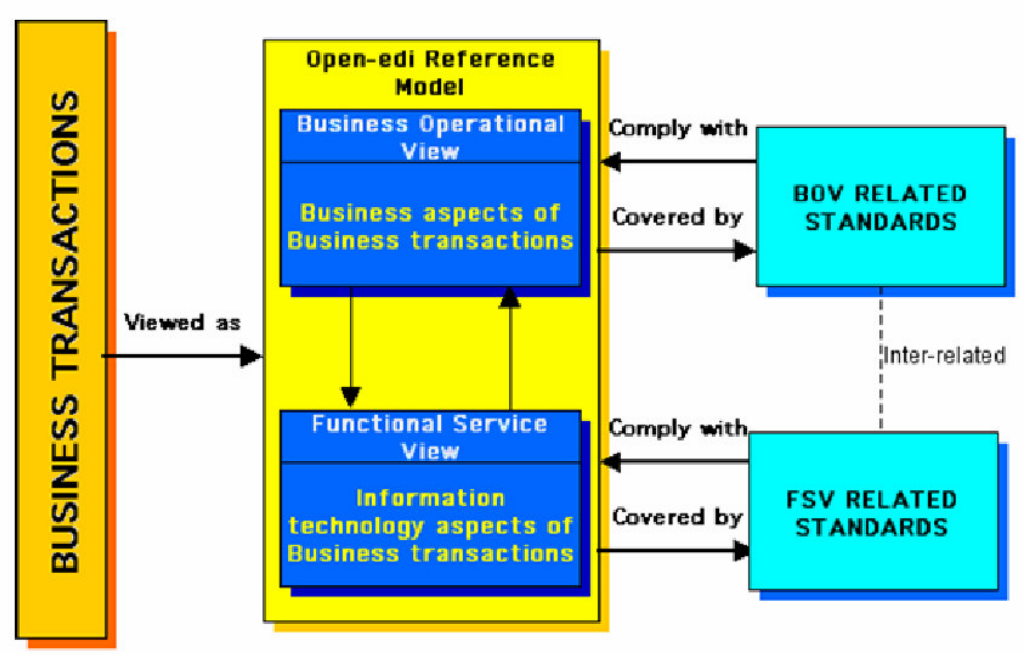


Figure 3 - 2: The open-edi reference model (from [UMM UG])

In this model, UMM is used in the Business Operational View (BOV), which is the most conceptual part of the open-edi framework. As said in [UMM UG], UMM is intended to provide "a perspective of business transactions limited to those aspects regarding the making of business decisions and commitments among persons, which are needed for the description of a business transaction".

3.2 The objectives of UMM

The objectives of UMM, as described in [UMM UG], are multiple. They can be summarized as follows:

- facilitate the analysis of e-business processes;
- help analysts to create business collaboration frameworks between trading partners;
- understand and formalize the dependencies between processes belonging to business partners in a particular domain;
- provide reusable process models and descriptions (often common to a whole sector of activity);
- maintain libraries (UMM Business Component Libraries) containing these reusable models and descriptions
- help the development of quality e-business platforms (ebXML systems for instance);
- create models that are comprehensible for both domain experts and software developers;
- create a methodology totally independent of the underlying implementation techniques (it means for example that mappings should be created to transform UMM data types in data types acceptable by programming languages);
- create models able to evolve in case of change in the requirements;
- extend UML (by using the mechanism of stereotypes) to support business concepts.

3.3 UMM in a nutshell

Figure 3-3 shows the relationships between the major UMM concepts. These concepts will be explained in point 3.4, which describes UMM step-by-step by presenting its modelling views.

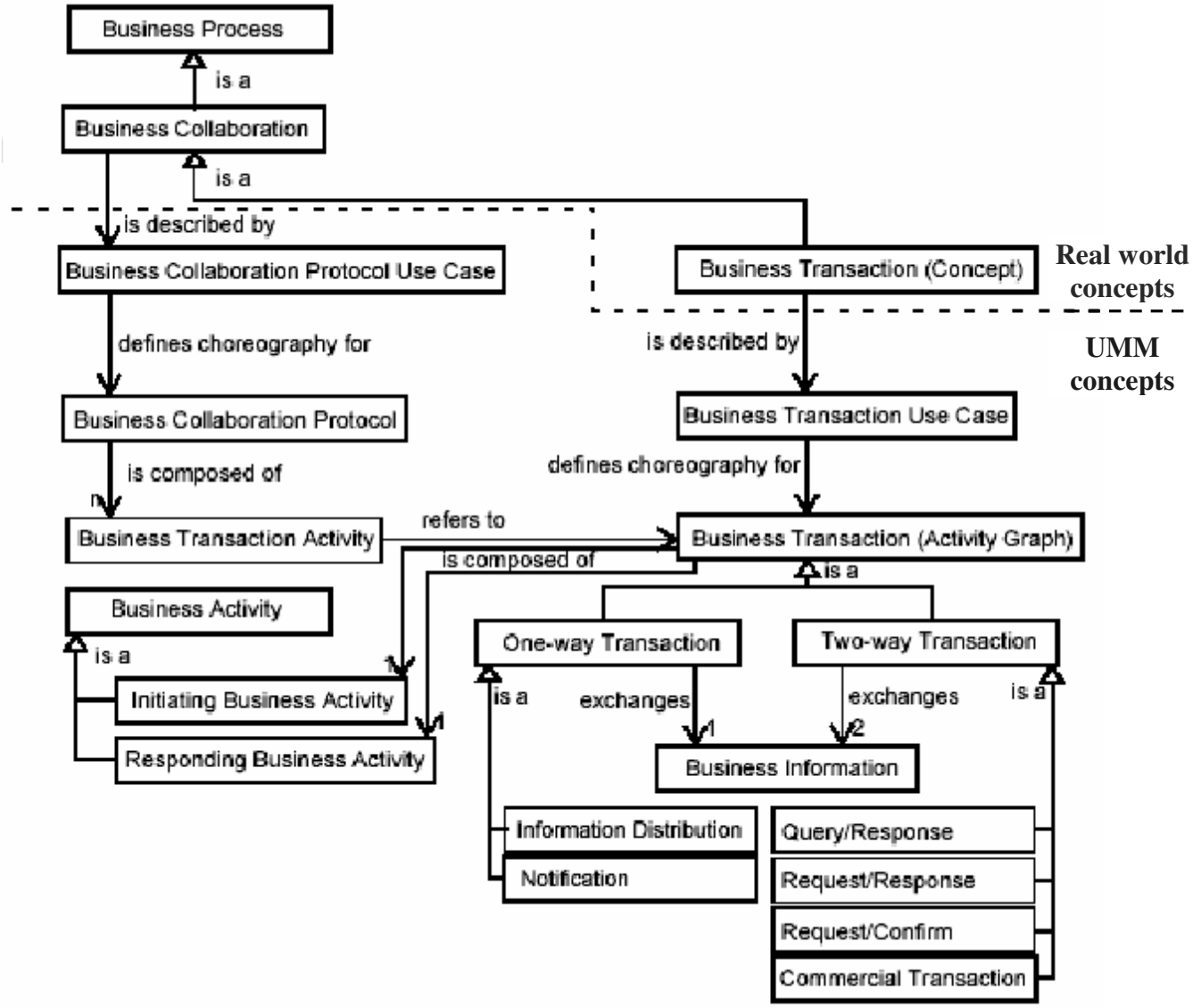


Figure 3 - 3: Relationships between UMM concepts (from [Hofreiter&al. 2004])

This diagram summarizes the hierarchy of UMM concepts.

3.4 The UMM views.

UMM is organized into four views, each describing a different business process perspective. The output of every view (a set of models and descriptions) is the input needed by the next view. So, the final models and descriptions are obtained through a workflow (a sequence of steps) approach. Modelling in UMM consists in documenting the views in filling in different worksheets and creating UML diagrams. We will see that, for each view, the meta-models are based on UML stereotypes, thereby extending UML.

The four views are:

- the *Business Domain View* (BDV);
- the *Business Requirements View* (BRV);
- the *Business Transaction View* (BTV);
- the *Business Service View* (BSV).

The three first views are called "work areas" because we have to fill in worksheets and create UML models. The fourth and last view, the BSV, is the result of the procedures performed in the three first views and is more technical (with regard to services and exchanged data) and. The study of the BSV is not relevant in the scope of this work since it is closer to the implementation phase and is not tightly linked to the modelling of business collaborations. For the same reasons, [UMM UG] does not describe the BSV.

Figure 3-4 shows the structure of UMM (BSV excluded).

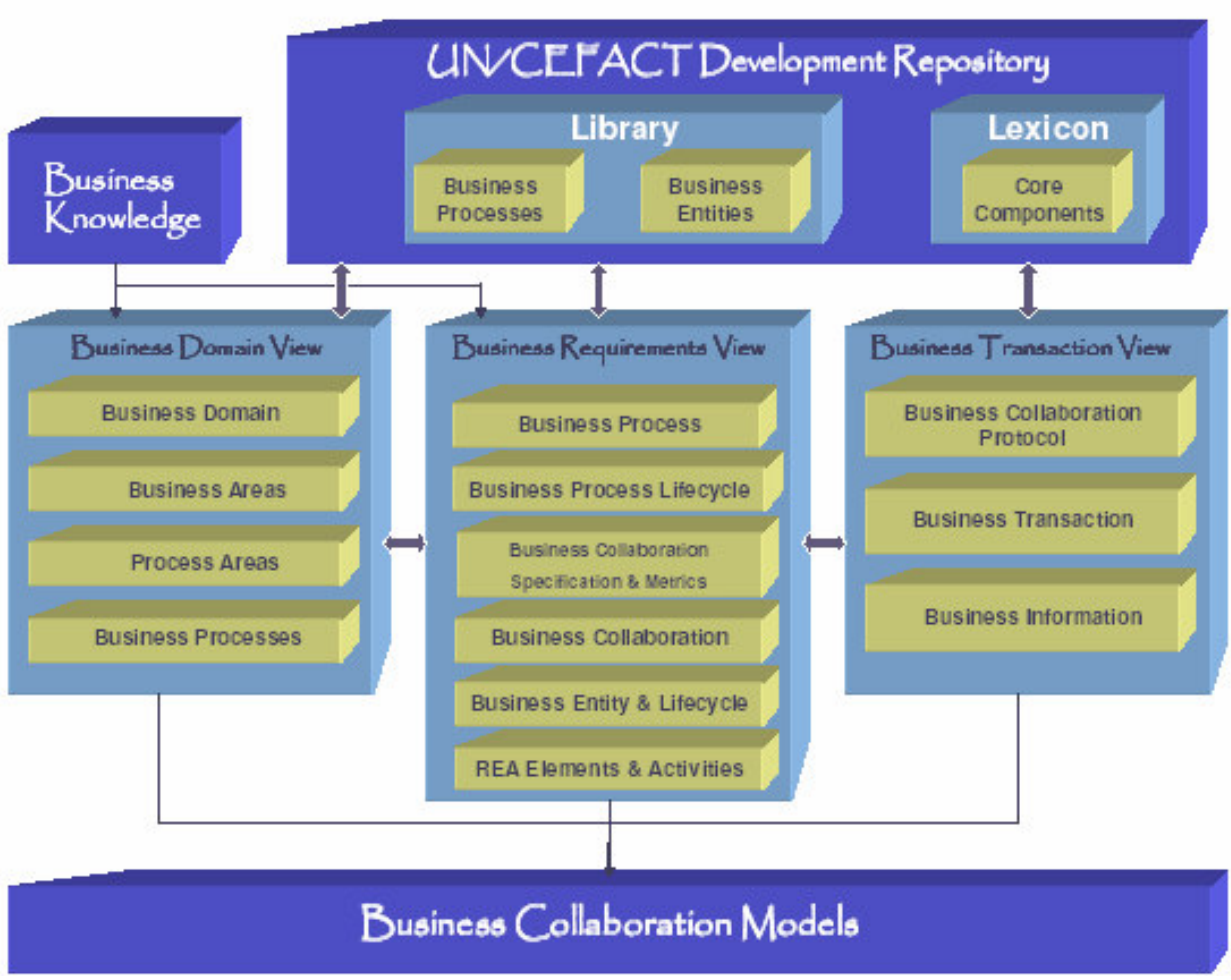


Figure 3 - 4: The structure of UMM (from [UMM UG])

This figure shows the worksheets present in each view. These worksheets, once filled in on the basis of business knowledge, are used to create models describing the collaboration. The figure also depicts the UMM libraries which store reusable descriptions of business processes and business entities. The lexicon contains core components used to build business objects. All these concepts (business process, business entity, business object, etc.) will be discussed in the description of the views.

3.4.1 The Business Domain View (BDV)

The Business Domain View, also called Business Operations Map (BOM), is used to identify the *Business Processes* that need an e-business collaboration. Once these collaborative *Business Processes* are identified, we check if it is possible to define them by reusing existing descriptions from the *UMM Business Component Libraries*. If not, we will have to create these descriptions by ourselves.

In the following, we will first provide the BDV meta-model, then we will describe the concepts used by this meta-model. Finally, we will explain what is done in the BDV work area.

BDV meta-model

The figures 3-5 and 3-6 show the relationships between the concepts of the BDV. The figure 3-7 further describes these concepts by showing their attributes. The UMM meta-models are described by UML class diagrams. As already mentioned, UMM extends UML by using UML stereotypes.

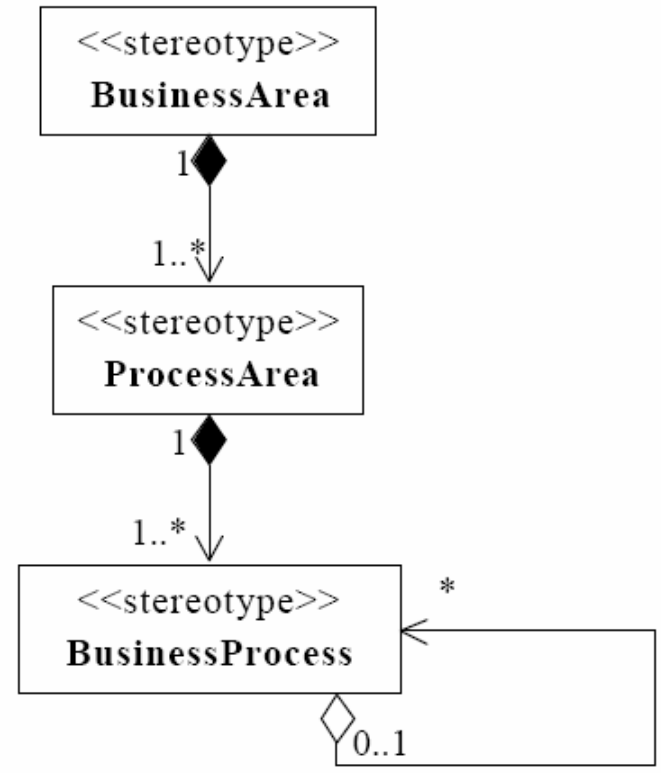


Figure 3 - 5: BDV meta-model: relationships part 1 (from [UMM-N090R10])

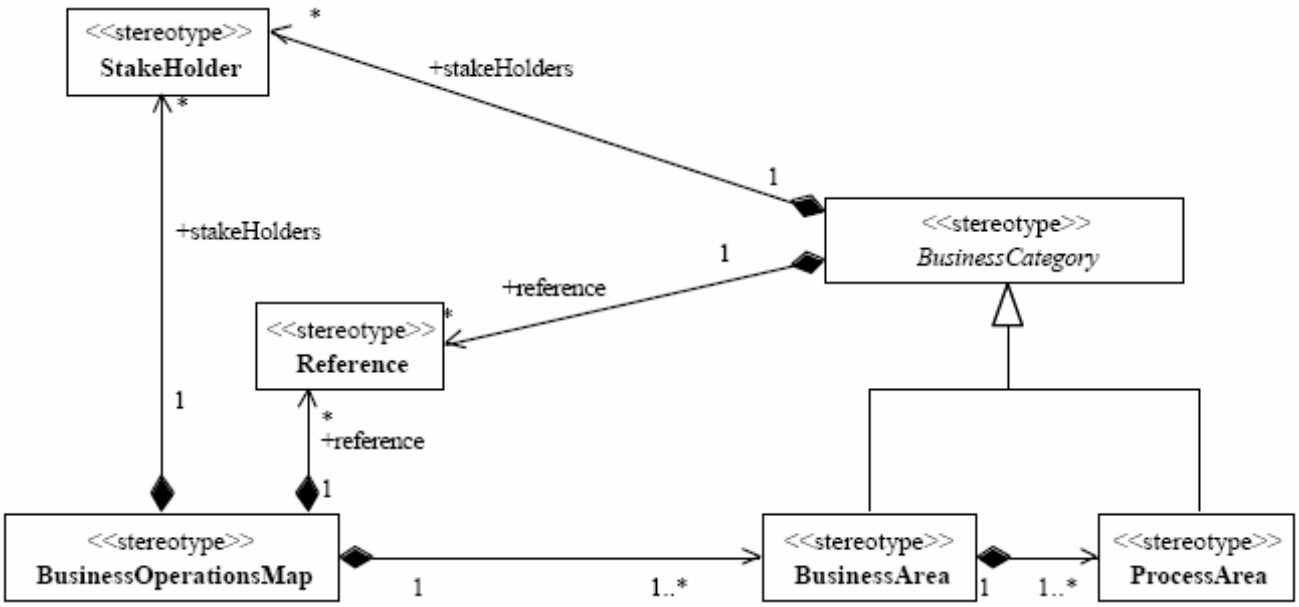


Figure 3 - 6: BDV meta-model: relationships part 2 (from [UMM-N090R10])

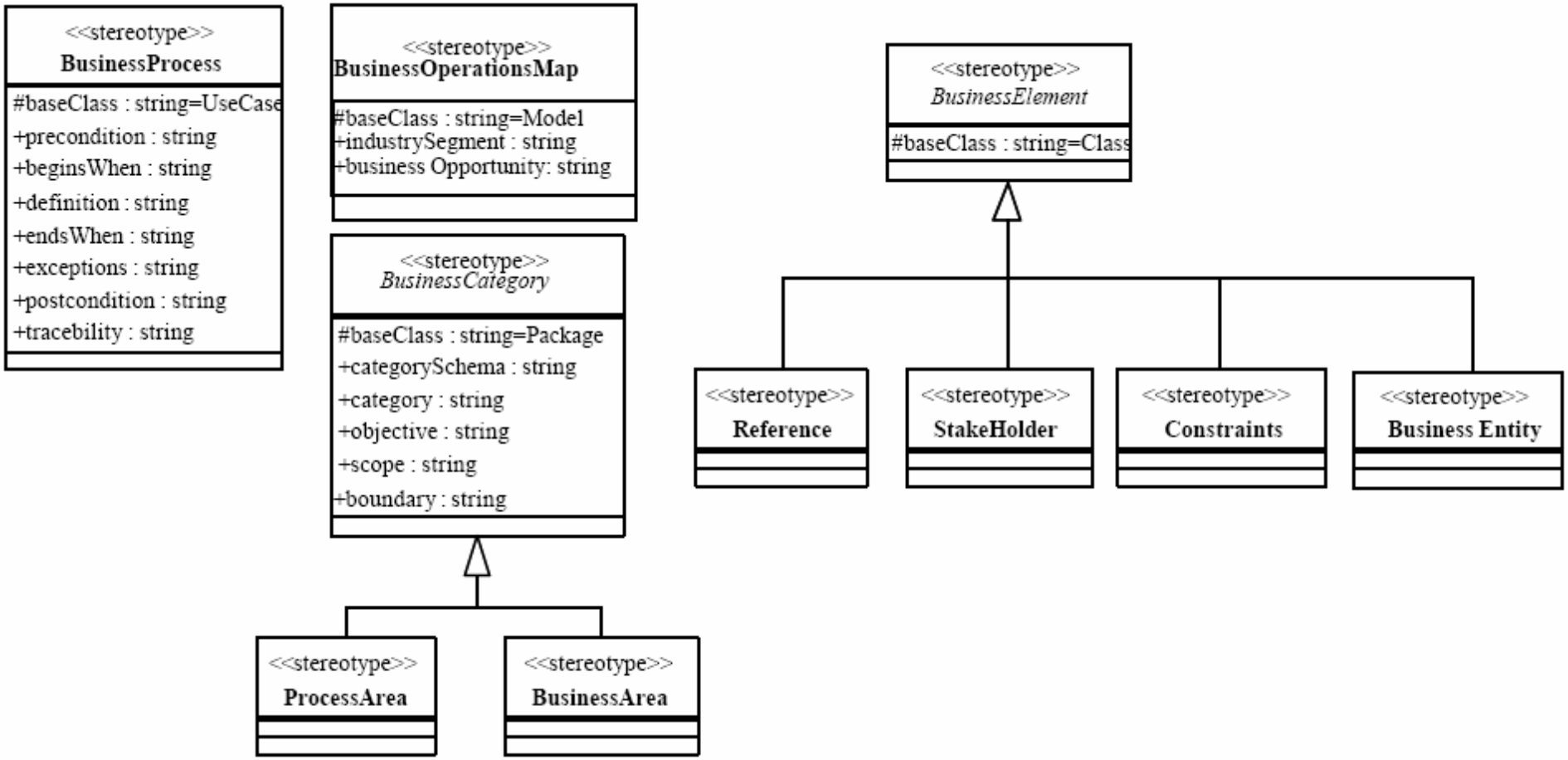


Figure 3 - 7: BDV meta-model: descriptions (from [UMM-N090R10])

BDV description

Here follow the definitions of the major concepts of the BDV. Because [UMM UG] is sometimes very ambiguous, it was necessary to take definitions from other sources in order to really understand UMM. The other concepts present in the meta-model (stakeholder, reference, constraint, business category, etc.) are less important in our discussion. The description of these elements, as well as the definition of all attributes, is available in Annex 2. We should also note that the *Business Entity* concept, which is fundamental in UMM, will not be defined here. Indeed, even if it is present in Figure 3-6, it is used nowhere in the diagrams showing the relationships between the BDV concepts (i.e. Figure 3-4 and 3-5). Actually, *Business Entities* are identified and described in the BRV (the second UMM view, which will be studied later). Therefore, the *Business Entity* concept should not appear in the description of the BDV. This is one of the inconsistencies present in UMM (UMM will be criticised at the end of this chapter). Here are the most important definitions (they can also be found in the glossary in Annex 1):

Business Domain Model (also known as the Business Operations Map)
(1): {In the BDV, we create the *Business Domain Mode*l which is a model of the application domain. The business domain is divided into *Business Areas*, *Process Areas* and *Business Processes*.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Area
(1): "An area of knowledge or activity characterized by a family of related systems; an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area" ([UMM-N090R10], Annex1_UMM_Glossary).
(2): {*Business Areas* are subdivisions of a *Business Domain Model*. They represent the structure of an enterprise or of a general framework (sector of activity for instance). Therefore, they are generally market segments (book market, shoe market, etc.) or big operational divisions (an insurance department for example). A *Business Area* can itself be divided into *Process Areas*.} (Our interpretation of the sometimes ambiguous [UMM UG])

Process Area
(1): {*Process Areas* either subdivide the *Business Domain Model* in a manner that is orthogonal to the categories chosen for the *Business Areas* or subdivide *Business Areas*. In the former case, *Process Areas* represent workflows that spread over several *Business Areas*. In the latter, they provide a second-level decomposition of the *Business Domain Model*. A *Process Area* can itself be decomposed into *Business Processes*.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Process
(1): "A *Business Process* is a use case that is used to gather requirements about business processes." ([UMM-N090R10], Chapter 8 MetamodelR10)
(2): "The means by which one or more activities are accomplished in operating business practices." ([UMM-N090R10], Annex1_UMM_Glossary)
(3): {A *Business Process* is defined as an organized group of related activities[1] that together create customer value. If all the activities are performed by one organization this leads to an intra-organizational business process. In B2B[2], the activities are executed by different organizations which collaborate to create value. UMM focuses on inter-organizational business processes and calls them *Business Collaborations*. A *Business Process* can divided into sub-*Business Processes*.} (Our interpretation of the sometimes ambiguous [UMM UG])

[1] An activity is an atomic unit of work. It is the lowest-level of functionality within a business process.
[2] B2B stands for "Business To Business". It is about e-business collaborations between enterprises. To be simple, e-business can be divided into two major categories: B2B and B2C ("Business To Customer", which is about e-business collaborations with private individuals)

Figure 3-8 depicts an example of structure of a Business Domain Model.

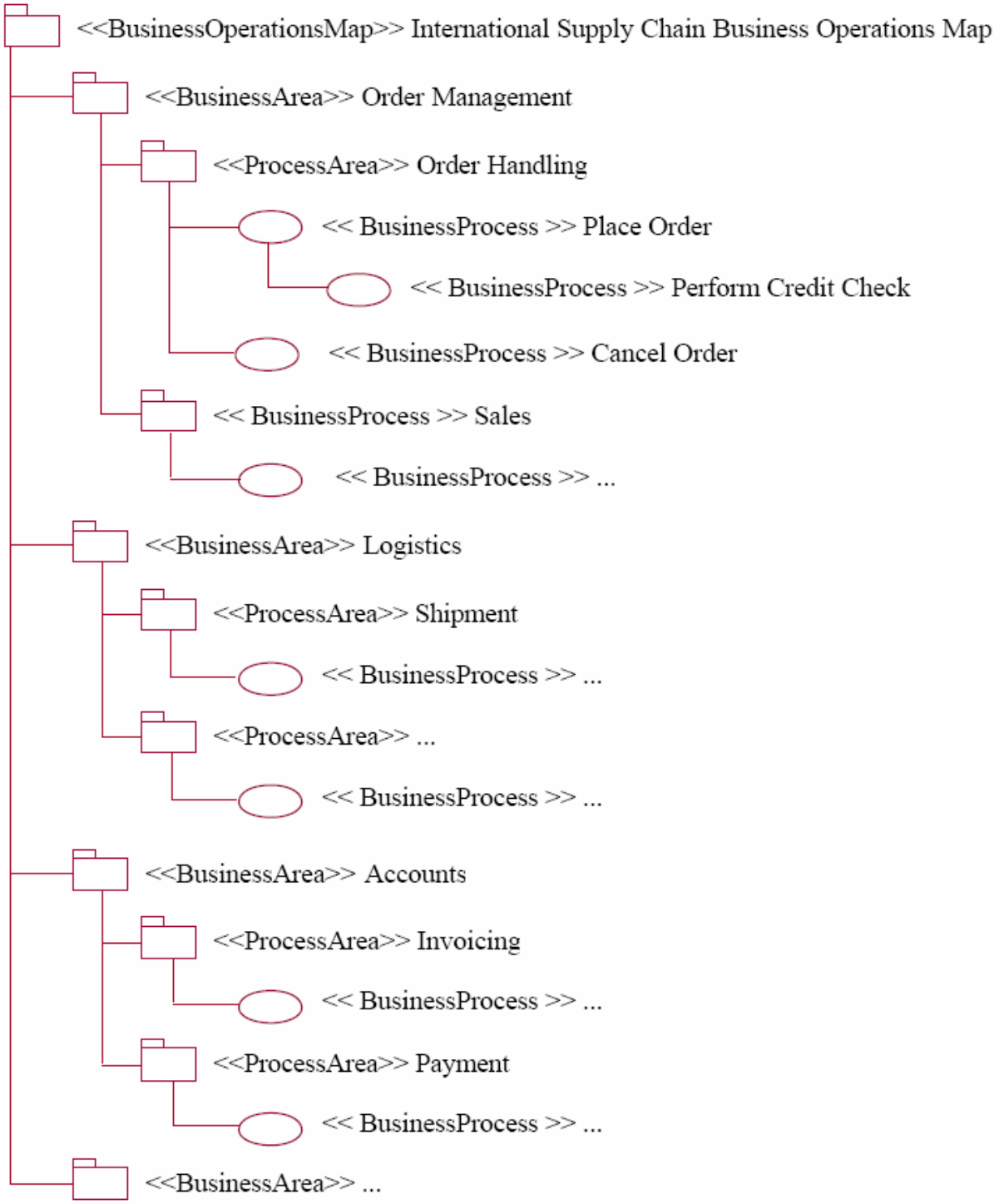


Figure 3 - 8: Example of a *Business Domain Model* (from [UMM-N090R10])

This example illustrates well the fact that UMM extends UML (as shown in the BDV meta-model). The BOM (which is a hierarchy of *Business Areas*, *Process Areas* and *Business Processes*) extends the UML package diagram and the *Business Process* extends the UML use case diagram.

BDV Work area

The BDV worksheets help to formalize the application domain. We identify the entities and organize the domain concepts. We have to highlight the terminology, the actors and the processes in which these actors are involved.

We can reuse an existing reference model for the sector of activity under study. If we do so, it is easier to achieve interoperability between partners.

In the BDV, UMM allows us to:

- understand the domain structure and the dynamics without thinking about software implementation;
- describe the domain in a language that domain experts, business analysts and software developers understand;
- identify the business stakeholders;
- subdivide the domain and, in doing so, allow an iterative modelling approach.

All the information we collect in the BDV is obtained by interviewing the business experts. The figure 3-9 shows the steps, the worksheets and the models (diagrams) in the BDV. Examples of these worksheets and models will be provided in the case study at the end of this section.

Steps	Artifacts	
	Section / Worksheet Name	Diagrams
1. Identify and Describe Business Area	4.2.1 / Describe Business Domain Model 4.2.2 / Describe Business Area	
2. Identify and Describe Process Area(s)	4.2.3 / Describe Process Area(s)	Business Area/Process Area Package Diagram
3. Identify Business Process(es)	4.2.4 / Identify Business Process(es)	Package diagram identifying and categorizing Business Processes within Business and Process Areas currently available in a Li-brary(Repository)
4. Identify Business Processes from the Business Process Library		BDV Use Case Diagrams for Library supported Business Processes
5. Identify and Finalize Business Processes and Partners		Final BDV Use Case Diagram Using Processes from Library (Processes and Partners Identified)

Figure 3 - 9: Steps, worksheets and models in the BDV (from [UMM UG])

3.4.2 The Business Requirement View (BRV)

The BDV has allowed us to identify the *Business Processes* we have to model. These *Business Processes* are collaborative business processes for which no description exists in the *UMM Business Process Library*. In the BRV, we will begin describing these processes.

The BRV defines how the Business Processes we are interested in happen in the real business world. It allows us to understand the execution, the inputs, the outputs, the constraints, and the boundaries of the studied *Business Processes*. This view uses the language of the domain experts (i.e. people who evolve in the studied business domain).

BRV Meta-model

Figure 3-10 shows the relationships between the concepts of the BRV. Figure 3-11 describes each of these concepts by listing its attributes. It is curious to notice that some concepts appear in Figure 3-11 but do not in Figure 3-10. This is one of the inconsistencies present in UMM (UMM will be criticised at the end of this chapter).

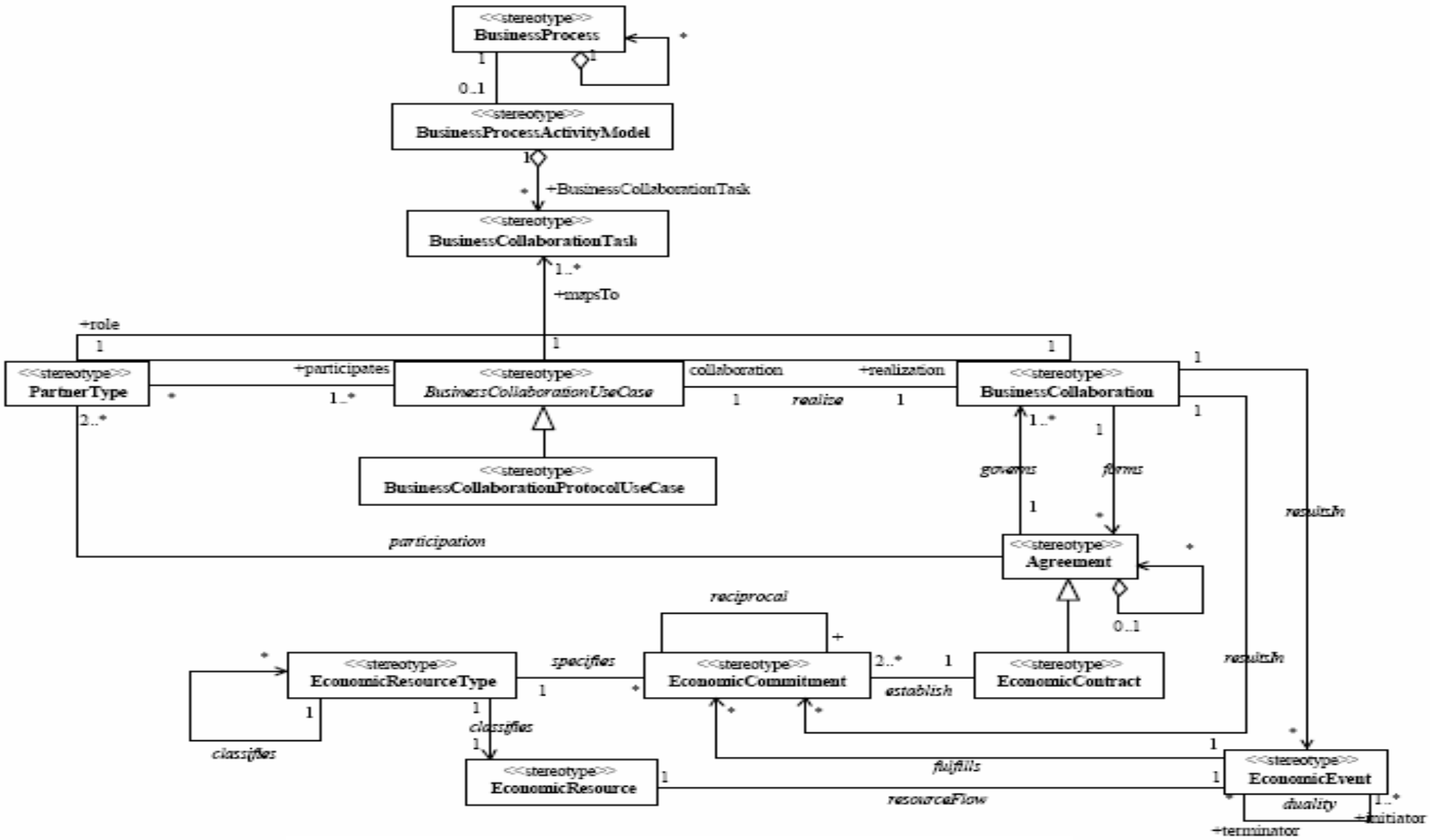


Figure 3-10. BRV meta-model: relationships (from UMM-N090R10])

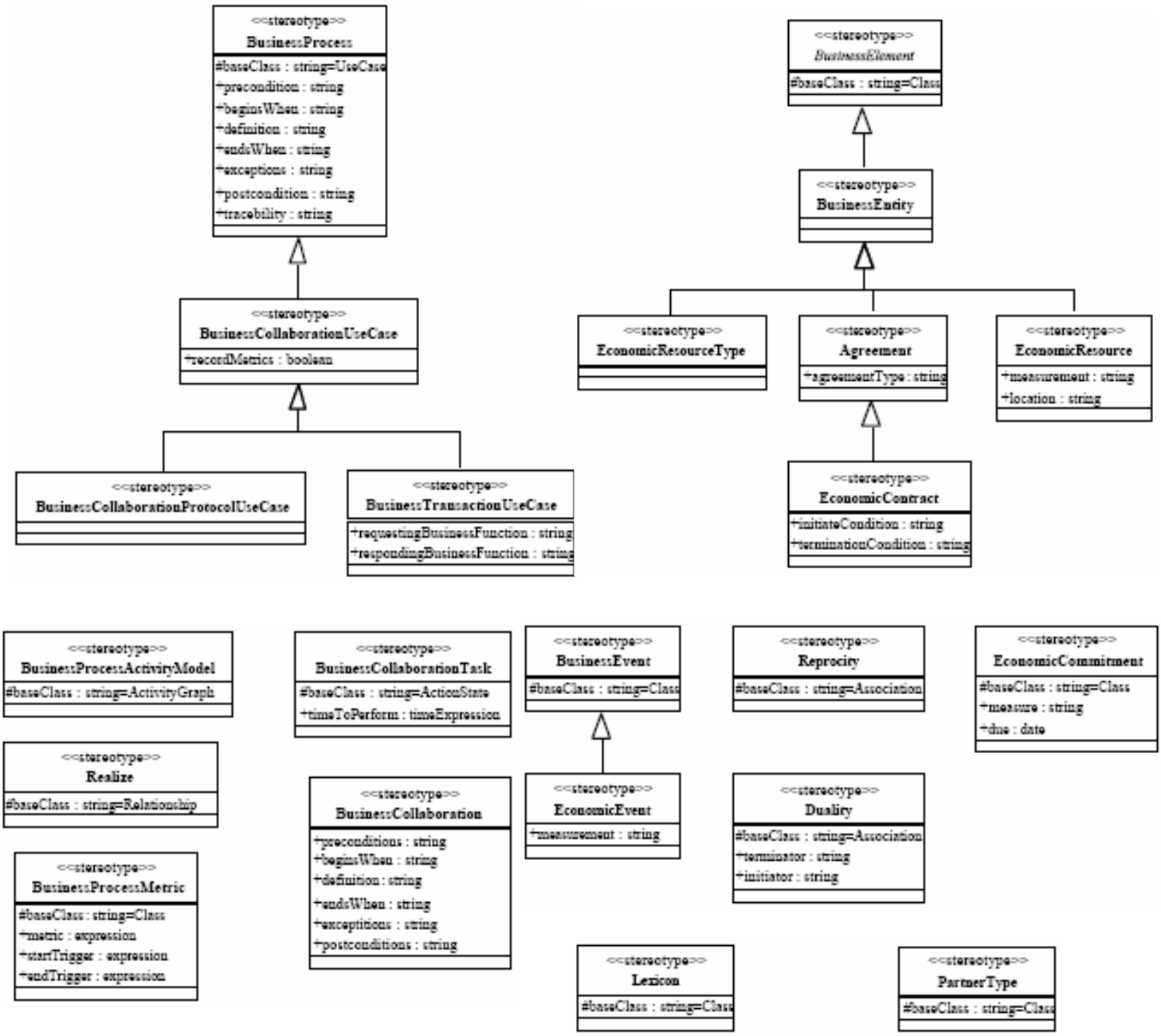


Figure 3 - 10: BRV meta-model: descriptions (from UMM-N090R10])

BRV description

Here follow the definitions of the major concepts of the BRV. Because [UMM UG] is sometimes very ambiguous, it was necessary to take definitions from other sources in order to really understand UMM. The other concepts present in the meta-model (BusinessCollaborationTask, Lexicon, Duality, Realize, etc.) are less important in our discussion. The description of these elements, as well as the definition of all attributes, is available in Annex 2. Here are the most important definitions (they can also be found in the glossary in Annex 1):

Business Collaboration
(1): "A *business collaboration* model specifies the input and output relationships between *business collaboration use cases* and Agents." ([UMM-N090R10], Chapter 8 MetamodelR10)
(2): {A *business collaboration* activity is a predefined set of activities of partners that is initiated by a partner to accomplish an explicitly shared business goal and terminated upon recognition of one of the agreed conclusions by all the involved partners. *Business collaboration* activities are specified by a business analyst as business processes, requirements and business object flow graphs that define the choreography of atomic business processes, referred to as *Business Transactions*.} (Our interpretation of the sometimes ambiguous [UMM UG])
(3): "A *business collaboration* is performed by two (binary collaboration) or more (multi-party collaboration) business partners. A *business collaboration* might be complex and involve a lot of activities between business partners. However, the most basic *business collaboration* is a binary collaboration realized by a request from one side and an optional response from the other side. This simple collaboration is called *business transaction*." [Hofreiter&al. 2004]

Business Collaboration Use Case
"A *business collaboration use case* is an abstraction for a *business collaboration protocol use case* and a *business transaction use case*. The abstraction permits the reuse of the business collaboration realization relationship. A completed use case assumes that some one "thing" of "measurable value" be created either as a service performed or a product created." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Collaboration Protocol (does not appear in the meta-model but is used in the work area)
(1): "A *business collaboration protocol* choreographs one or more *business transaction* activities." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {A *Business Collaboration Protocol* is composed of *Business Transactions* (each having states, preconditions, and postconditions). They are represented by an activity graph. The *Business Collaboration Protocol* defines the transitions between *Transaction activities*. It is based on the states of the entities. Therefore, it defines the choreography of the whole collaboration. Each activity from the *Business Collaboration Protocol* is a *Transaction activity* and is further detailed by a *Transaction*. For each *Transaction Activity*, there is exactly one *Transaction* (and vice-versa). These two notions are synonyms in the business world but are modelled differently.} (Our interpretation of the sometimes ambiguous [UMM UG])
(3): "Since UMM is based on UML, it uses the concept of use cases to capture requirements. In case of a complex business collaboration, the requirements are described in a so-called *Business Collaboration Protocol Use Case*. These requirements lead to a choreography of activities in order to create the customer value. The activity graph representing this choreography is called *Business Collaboration Protocol.* Each activity shown in a *Business Collaboration Protocol* refers to exactly one *Business Transaction*. Therefore, each activity of the *Business Collaboration Protocol* is called a *Business Transaction activity*." [Hofreiter&al. 2004]

Business Collaboration Protocol Use Case
"A *Business Collaboration Protocol Use Case* is used to gather requirements for e-business collaboration protocol specifications." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Transaction Use Case
"A *Business Transaction Use Case* is used to gather requirements for *Business Transaction* specifications." ([UMM-N090R10], Chapter 8 MetamodelR10)

Partner Type
"A *Partner Type* is an actor in a *Business Collaboration Use Case*. Partner types are manufacturer, distributor, retailer, end user, carrier and financier. " ([UMM-N090R10], Chapter 8 MetamodelR10)

Agreement
"An *Agreement* is an arrangement between two partner types that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific *Economic Commitments*." ([UMM-N090R10], Chapter 8 MetamodelR10)

Economic Contract, Economic Commitment, Economic Event, Economic Resource, Economic Resource Type
These terms come from the REA ontology which will be explained in details in the BRV work area just after this part.

Business Entity
(1): "Something that is accessed, inspected, manipulated, produced, and so on in the business." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): "*Business Entity* is an abstraction for any artifact that is important in the execution of a business collaboration." ([UMM-N090R10], Chapter 8 MetamodelR10)

Examples of *Business Entities* include an invoice, an order for a product, etc.

BRV Work area:

We express the requirements by referring to the *Business Entities* (such as an order, a merchandise, etc.) that are affected in a business collaboration.

The specifications (preconditions, postconditions) of a *Business Process* and the collaborations it contains are expressed in terms of the states of the *Business Entities*. For instance, a *Business Entity* "order" could be in several states including "pending" or "executed". The states and the transitions between states are clearly described. We must also define the events triggering the transitions. An example of such an event could be the delivery of a product.

A collaboration is a set of predefined activities belonging to the business partners. That set of activities is initiated by one of the partners to achieve a shared objective. It is ended when some accepted conclusion arises. That conclusion must be agreed by all participating partners. The collaboration activities are formally described in terms of activities, requirements and object-flow graphs. These graphs are called *Business Transactions* and define the choreography of the studied *Business Processes*. It is possible to reuse an existing collaboration pattern.

Concretely, in the BRV, we divide the collaboration in three phases: planning / identification, negotiation, and actualization / post-actualization.

The planning / identification phase consists in finding the activities involved in the seeking of means to acquire or sell something, and in the identification of potential partners to reach that goal. Examples of such activities could be the request for / the sending of a catalogue.

The negotiation phase consists in identifying the activities involved in the contacting process among the business partners. Examples of such activities could be the sending of an offer and the sending of the contract acceptance.

The actualization / post-actualization phase consists in identifying the activities involved in the execution of economic events (economic resources transfers) and in the follow-up of the transaction. Examples of such activities could be the sending of a shipment notice, the sending of a bill, and the sending of the warranty invocation.

In the course of these different phases, we describe all the elements that take part in the collaboration: the partners, the resources, the event types, location types, the commitments, the agreements (contracts for example) and the exceptions that could prevent the good execution of the collaboration.

As said before, an UMM work area consists in filling up worksheets and making diagrams. In the BRV, we begin with the REA worksheet and its associated class diagram. Since REA is the base of the BRV, it is important to explain its basic principles.

REA[1], which stands for "Resources Events Agents", is a business collaboration ontology. There is no consensus on the definition of the term « ontology ». [UMM UG] considers an ontology is a "specification of a conceptualization"[2]. REA is a specification of the declarative semantics involved in a business collaboration (and more generally in a business process). As a consequence, the REA definitions are useful to build e-business systems. In UMM, the REA definitions are used to characterize collaborations involving economic exchanges that are closely synchronized.

The REA ontology mainly consists of a UML class diagram representing the concepts underlying a business collaboration and the associations between these concepts. Figure 3-12 depicts the basic structure of REA.

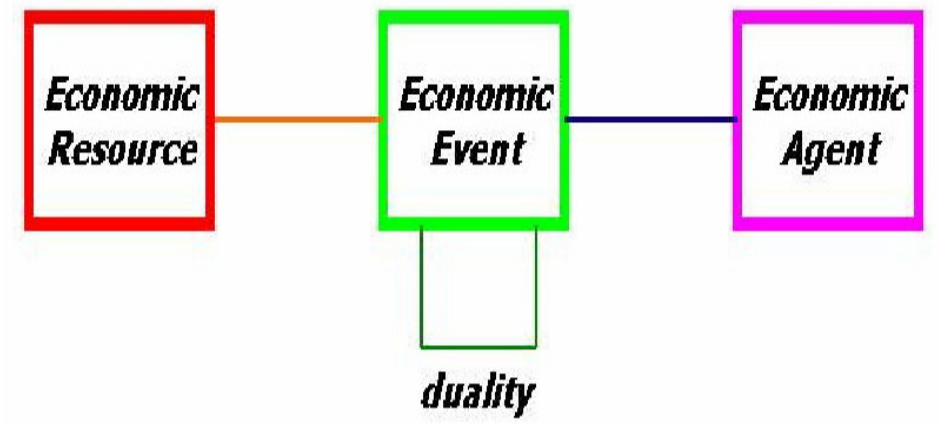


Figure 3 - 11: Basic structure of REA (from [UMM UG])

A business collaboration implies two dual *Economic Events*. Each of these events is triggered by a different *Economic Agent* (i.e. a trading partner) and consists in transferring the ownership of an *Economic Resource* (a product, a service, money, etc.) to another *Economic Agent*. For instance, a supplier could provide a customer with a bottle of whine in exchange for money. Figure 3-13 shows a REA model representing this situation.

[1] REA was first published in [McCarthy 1982]
[2] This definition comes from [Gruber 1993]

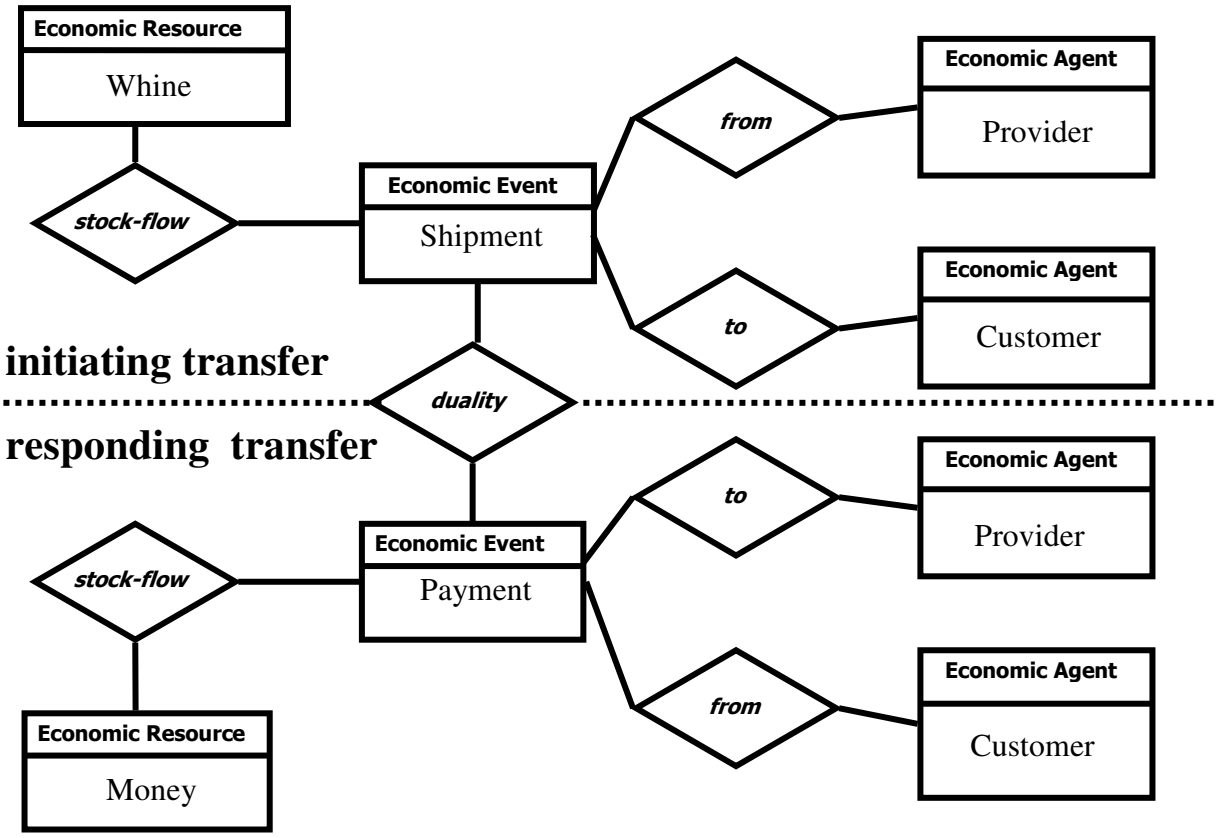


Figure 3 - 12: Example of REA model

The basic principles explained above are enough to represent a simple trading operation. However, long-term collaborations need more trust and should be more predictable. Therefore, it would be nice to add contractual principles to the basic REA. Figure 3-14 shows how UMM enrich the basic REA class diagram by adding the management of commitments.

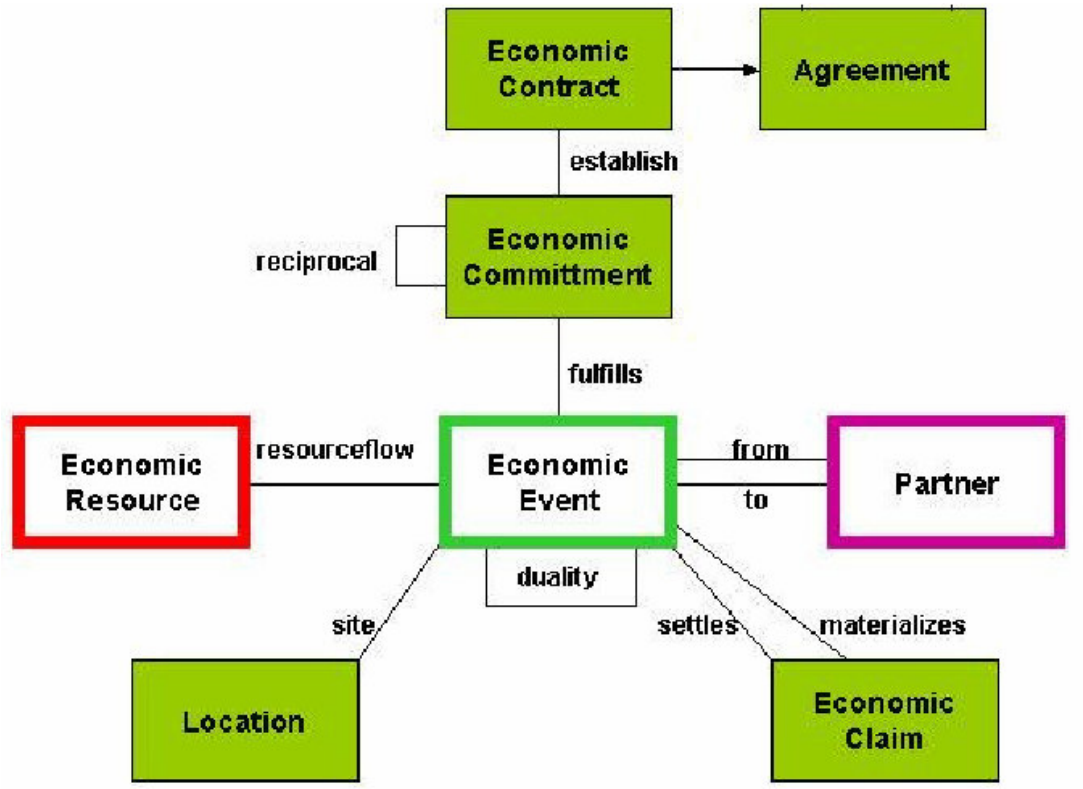


Figure 3 - 13: REA with the management of commitments (from [UMM UG])

An *Economic Contract* is an agreement (i.e. is a subtype of *Agreement*) between the trading parties. Each partner commits itself in advance to trigger an *Economic Event*. The *Economic Contract* describes the modalities of the future exchanges between the partners. *Economic Claims* are used when documentation of partially completed exchanges is needed (in the case of a payment in several times for instance). *Location* is used to identify the place where an *Economic Event* takes place.

The model depicted in Figure 3-14 is purely descriptive. It does not allow to specify control policies or collaboration patterns (which could be stored in libraries). To allow such specifications, types are needed. Figure 3-15 shows what the REA structure becomes when types are added.

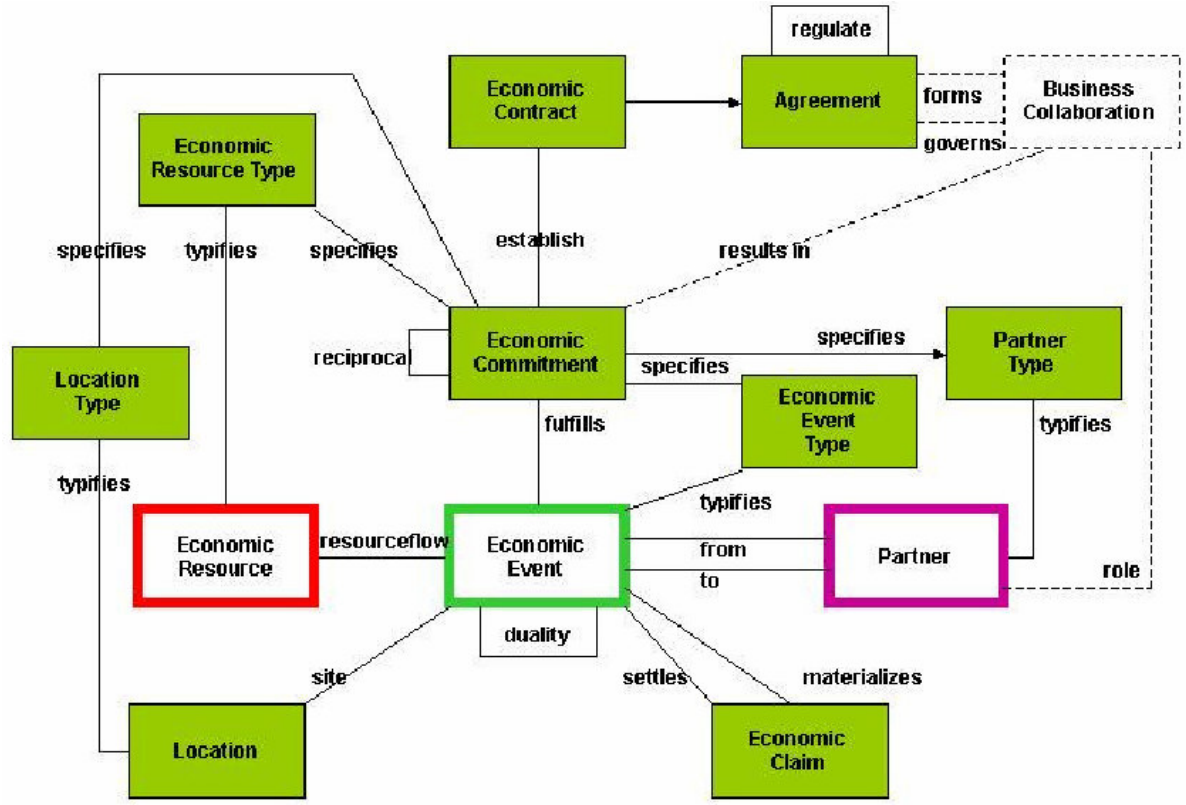


Figure 3 - 14: REA with the management of commitments and types
(from [UMM UG])

New classes were added in the REA class diagram in order to represent the types. For instance, the class *Economic Resource Type* could be used to give information about which family of whines (i.e. Bordeau, Bourgogne, etc.) a whine belongs to. The class *Economic Event Type* could be used to describe the kind (conditions, etc.) of transaction to be performed. The class *Partner Type* could be used to mention if the partner is a provider or a customer. The class *Location Type* is used to specify the kind of place where the economic event is to be performed.

This last figure gives a good idea of the main concepts behind the UMM *Business Requirements View*. However, not all classes shown in Figure 3-15 are used in the BRV meta-model. For instance, the classes *Location*, *Location Type* and *Economic Event Type* are not used in UMM. On the other hand, some classes (*Business Process* for instance) exist in the BRV meta-model but do not in Figure 3-15. The creators of UMM have only used REA as a basis for the BRV. They did not take REA in its entirety.

Once the REA diagram is done, we have to precisely specify the *Business Processes* and implied collaborations. For this, we describe and specify the *Business Collaborations* by dividing them in so called *Business Transactions*. Actually, we can identify two kinds of *Business Collaborations*: *Business Collaboration Protocols* and *Business Transactions*.

A *Business Collaboration Protocol* is composed of *Business Transactions* (each of which has states, preconditions and postconditions). The order of *Business Transactions* inside a same *Business Collaboration Protocol* is important. So, UMM proposes to represent *Business Collaboration Protocols* by an UML activity graph.

A *Business Transaction* is an atomic collaboration, that is, a collaboration that cannot be further decomposed in sub-level collaborations. A *Business Transaction* must match one of six standard transaction patterns. We will talk about this in the BTV description.

The figure 3-16 shows the steps, the worksheets and the models in the BRV. Examples of these worksheets and models will be provided in the case study at the end of this section.

Steps	Artifacts	
	Section / Worksheet Name	Diagrams
1. Describe REA Elements and Activities of the Business Process Phases	5.2.1 / REA Worksheet	
2. Describe each Business Process (from BDV) in more detail	5.2.2 / Business Process	BRV Use Case Diagram with all identified business processes and partners
3. Identify and describe Business Collaborations starting with a large collaboration and breaking it down to smaller business collaborations use cases which need to be further described until business transactions are identified and described	5.2.3 / Business Collaboration Specification 5.2.4 / Business Process Metric	
4. Define Business Collaborations	5.2.5 / Business Collaboration 5.2.6 / Business Proc-ess/Collaboration Lifecycle (Activity Model)	Business Process Activity Model Conceptual Business Information Model Business Process Use Case Business Collaboration Use Case
5. Identify and Describe Business Entities	5.2.7 / Business Entity 5.2.8 / Business Entity Lifecycle	

Figure 3 - 15: Steps, worksheets and models in the BRV (from [UMM UG])

3.4.3 The Business Transaction View (BTV)

The BTV describes the semantics of *Business Information Entities* and their exchange (i.e. flows) between the partners accomplishing a business activity. This view is based on what has been defined in the BRV (actually, the BTV is a refinement of the BRV). It uses the language of business analysts, which can be considered as an intermediate language between the one of domain experts and the one of software developers.

BTV Meta-model

The figure 3-17 shows the relationships between the concepts of the BTV. The figure 3-18 describes these concepts by listing its attributes.

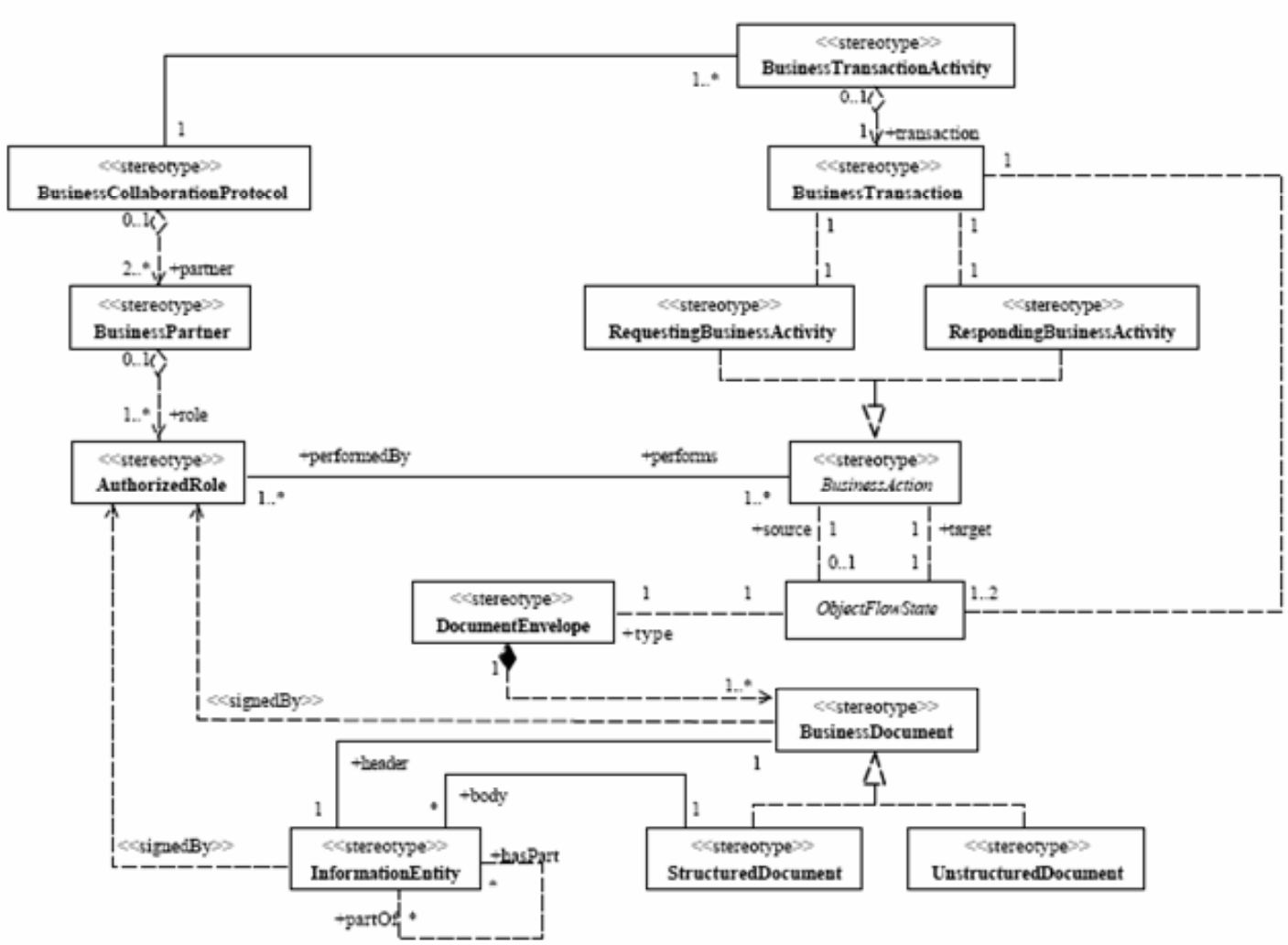


Figure 3 - 16: BTV meta-model : relationships (from [UMM-N090R10])

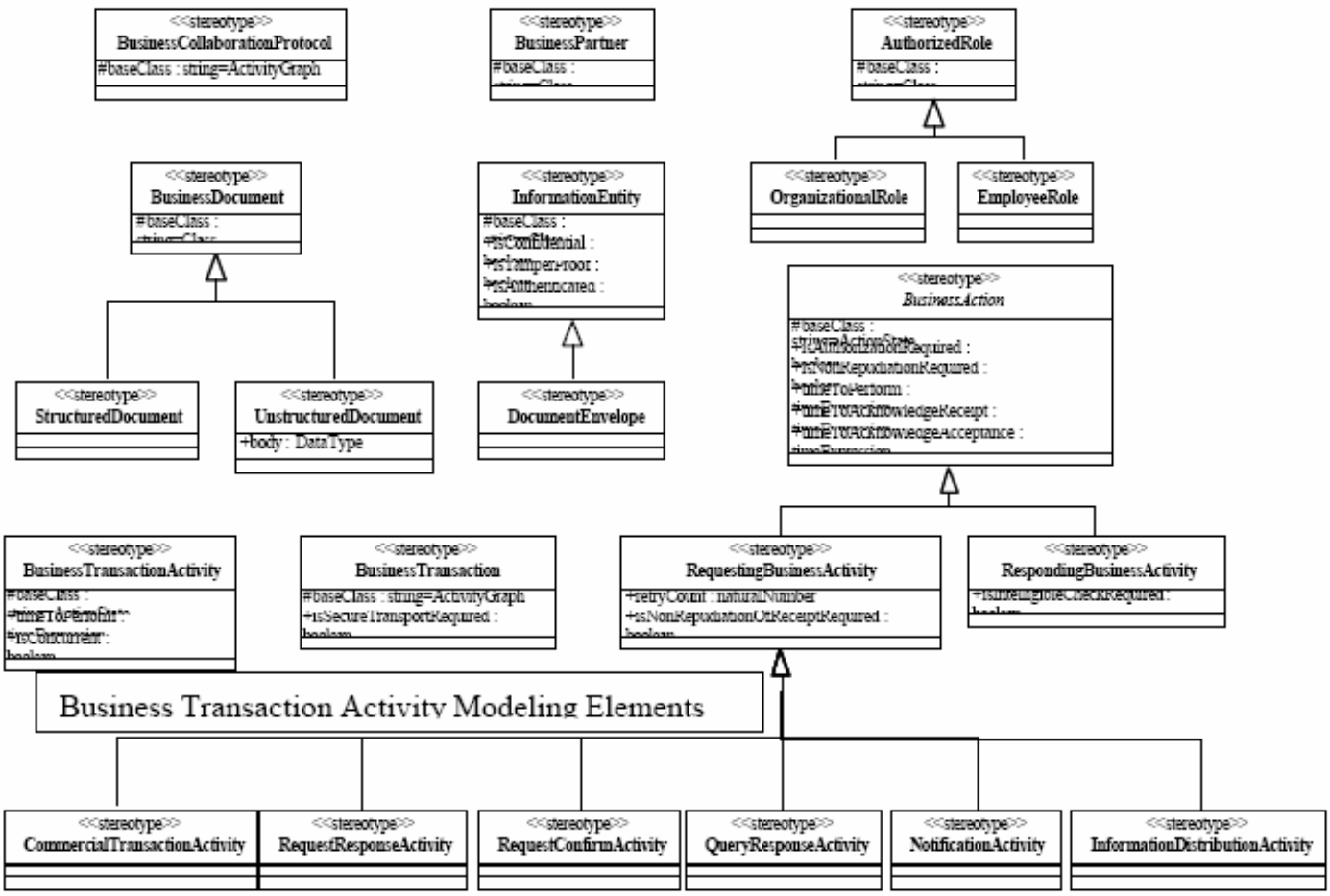


Figure 3 - 17: BRV meta-model: descriptions (from [UMM-N090R10])

BTV description

Here follow the definitions of the major concepts of the BTV. Because [UMM UG] is sometimes very ambiguous, it was necessary to take definitions from other sources in order to really understand UMM. The other concepts present in the meta-model are less important in our discussion. The description of these elements, as well as the definition of all attributes, is available in Annex 2. Here are the most important definitions (they can also be found in the glossary in Annex 1):

Business Transaction Activity
"A *Business Transaction Activity* is a *Business Collaboration Protocol Activity* that executes a specified *Business Transaction*." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Transaction
(1): "A *Business Transaction* is a set of business information and business signal exchanges between two business partners that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded. *Business Transactions* can be formal as in the formation of on-line offer/acceptance business contracts and informal as in the distribution of product announcements." ([UMM-N090R10], Chapter 8 MetamodelR10)
(2): {A *Business Transaction* is an atomic collaboration. There exist six transaction patterns. Each *Business Transaction* must belong to one of these patterns. A *Business Transaction* involves sending business information from one partner to the other and an optional reply.} (Our interpretation of the sometimes ambiguous [UMM UG])

(3): "The most basic *Business Collaboration* is a binary collaboration realized by a request from one side and an optional response from the other side. This simple collaboration is a unit of work that allows roll back to a defined state before it was initiated. Therefore, this special type of collaboration is called *Business Transaction*. The requirements of a *Business Transaction* are described by a *Business Transaction Use Case*. The requirements lead to a choreography of the *Business Transaction*. The resulting activity graph is what is really called *Business Transaction* in UMM. One might argue that *Business Transaction Activity* and *Business Transaction* represent the same concept. Since different UML elements - an activity and an activity graph - are required in the UML notation, these concepts are distinguished in UMM. The activity graph of a *Business Transaction* is always composed of two business activities, an initiating business activity performed by the initiator and a reacting business activity performed by the other business partner. In a one-way transaction, business information is exchanged only from the initiating business activity to the reacting business activity. In case of a two-way transaction, the reacting business activity returns business information to the initiating business activity. In UMM we distinguish two one-way transactions (which are two different *transaction patterns*) - *notification* and *information distribution*- and four two-way transactions (which are four different *transaction patterns*) - *query/reponse*, *request/confirm*, *request/response* and *commercial transaction*. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi." [Hofreiter&al. 2004]

Requesting Business Activity
"A *Requesting Business Activity* is a business activity that is performed by a partner role requesting business service from another *Business Partner* role." ([UMM-N090R10], Chapter 8 MetamodelR10)

Responding Business Activity
"A *Responding Business Activity* is a business activity that is performed by a partner role responding to another *Business Partner* role's request for business service." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Action
"The state of a *Business Transaction* is defined by reciprocal *Business Actions* executed by an authorized role. This is an abstract class that is not a stereotype." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Document
A *Business Document* is a set of information that has business significance and is exchanged between *Business Partners*. It is built by customizing and combining *Business Information*.

Information Entity
"An *Information Entity* realizes structured Business Information that is exchanged by partner roles performing activities in a business transaction. *Information Entities* include or reference other information entities through associations. A secure *Information Entity* is an *Information Entity* with security controls. " ([UMM-N090R10], Chapter 8 MetamodelR10)
Cf. the definition of Business Information

Business Information
{In a collaborative environment, partners must exchange *Business Information* to know the current states of the *Business entities*. Therefore, *Business Information* exchanges provoke changes in the states of *Business Entities*. The *Business Information* must indicate all the entities that change state following the exchange. Moreover, the *Business Information* contains a header with more general information (i.e. independent of the *Business Entities*).} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Object (does not appear in the meta-model but is used in the work area; it seems to be a concept similar to *Information Entity*)
{*Business Information* is manifested by *Business Objects*. A *Business Objec*t is a reusable class representing a particular business concept. In combining these objects, we are able to create business information structures. There exists a library containing a large amount of reusable *Business Objects*.}
(Our interpretation of the sometimes ambiguous [UMM UG])

BTV Work area

In the BRV, we already have all the descriptions of the studied collaboration and all the known transactions are included. From these descriptions, we must now define the choreography of *Business Transactions* inside the collaboration. This is done by an UML activity graph, called *Business Collaboration Protocol*.

The *Business Collaboration Protocol* describes the transitions between transaction activities, thereby defining the choreography of the whole collaboration. It is based on the *Business Entity* states.

Each UML activity inside the protocol is a *Business Transaction Activity*, which is further described by a *Business Transaction* (which is itself an UML activity graph). For each business activity, there is a transaction and vice-versa (the cardinality is one-to-one). These notions are synonyms in the world of business but are modelled differently in UMM.

A *Business Transaction* is an atomic process between partners. It implies the sending of information from a partner to the other and an optional reply. A *Business Transaction* comprises an initiating activity (a request from the initiating partner) and a responding activity (from the responding partner).

The initiating activity provides business information in output. That information is used as input by the responding activity. A Business Transaction must match one of six predefined transaction patterns. Figure 3-19 shows these patterns and the way we can find which pattern the *Business Transaction* belongs to.

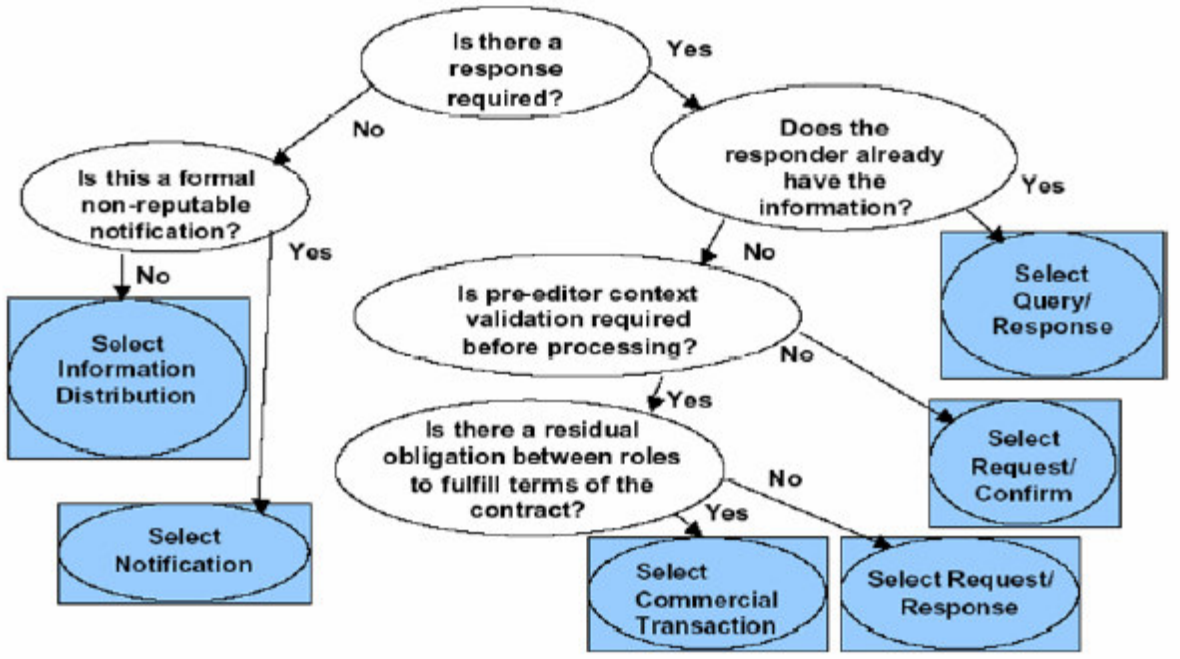


Figure 3 - 18: The Business Transaction Patterns (from [UMM UG])

The goal of a *Business Transaction* is to trigger (in accordance with the business requirements) a state transition on a *Business Entity*.

In a collaborative environment, partners have to exchange *Business Information* to know the current state of the entities. Therefore, the business information exchange provokes the state transitions on the *Business Entity*.

The *Business Information* must mention all the *Business Entities* that are changing their state because of the exchange. Moreover, the *Business Information* also contains a header with general information (information that does not refer to *Business Entities*).

All these pieces of information are manifested by so called *Business Objects*. A *Business Object* is a reusable class representing a particular business concept. In combining these objects, we are able to create business information structures.

The *Business Objects* are reusable because they are not specific to any transaction. There exists a library containing a large amount of reusable *Business Objects*. When we have to model information structures in the collaboration, we have to select the appropriate *Business Object* in this library and customize them in function of the studied transaction.

The customization of a *Business Object* can be divided in two tasks. First, we have to establish the relationships between the chosen objects. The context in which two objects are associated is given by an association role. Secondly, we have to choose the object attributes in a list in function of the studied context (i.e. we choose the needed attributes to change the state of a *Business Entity*).

The figure 3-20 shows the steps, the worksheets and the models in the BTV. Examples of these worksheets and models will be provided in the case study at the end of this section.

Steps	Artifacts	
	Section / Worksheet Name	Diagrams
1. Define a Business Collaboration Protocol (object state flow diagram) for each business collaboration use case (built by business transaction activities)	6.2.1 / Business Collaboration Protocol (Activity Model)	Business Collaboration Object Flow Diagram
2. For each Business Transaction activity define a business transaction activity graph. Identify requesting information and optional responding information	6.2.2 / Business Transaction	Use Case Diagram Business Transaction Object Flow Diagram
3. Create class diagrams by re-using existing information structure	6.2.3 / Business Information	Final Business Information Models

Figure 3 - 19: Steps, worksheets and models in the BTV (from [UMM UG])

3.4.4 The Business Service View (BSV)

The BSV defines the services, the agents, and the messages that are needed to establish a Business Collaboration. This view uses the technical terms of the software developers. This view is obviously closer to the implementation than to the conceptual work. It is less developed than the others views since it is not a work area. [UML UG] just provides a short description of it without going into the details. The study of BSV is not relevant in the context of this thesis because it is not directly linked to enterprise modelling nor e-business collaborations modelling.

3.5 Comments on the modelling approach

The modelling approach in UMM is decisively top-down (even if the construction of the Business Information by combining existing Business Objects is bottom-up). We begin by modelling the environment of the processes and therefore, the environment of the collaborations. Then we describe these collaborations in more details, defining the *Business Entities* involved, their states, their lifecycle, etc. Afterwards, we divide the collaborations into transactions and show how the business information is exchanged during business transactions. The following diagram, already presented in Figure 3-3, is a reminder of the relationships between the major concepts of UMM:

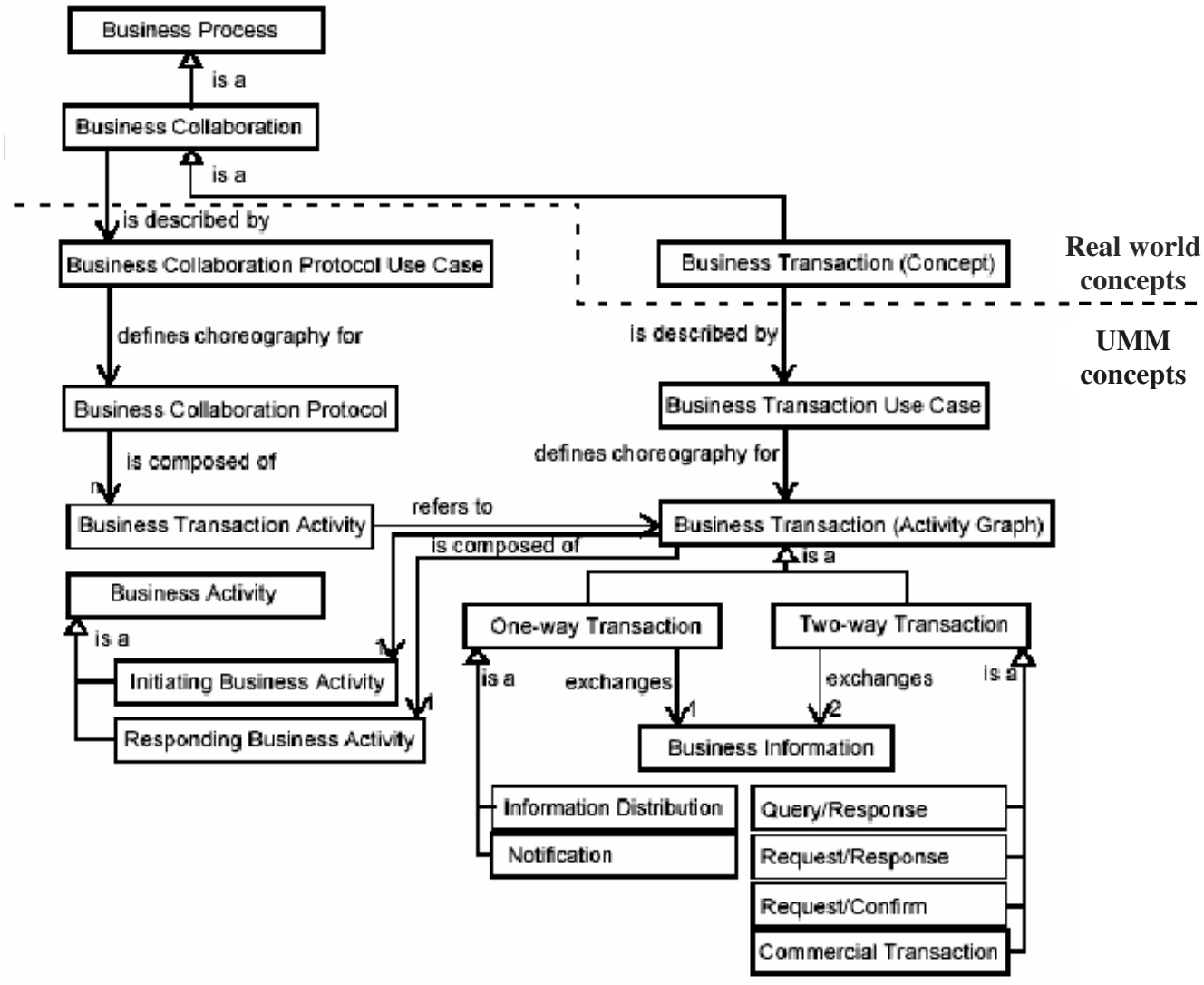


Figure 3 - 20: Relationships between UMM concepts (from [Hofreiter&al. 2004])

That top-down approach has several advantages, it allows us:

- to create reusable, easily-modifiable models;
- to manage loosely coupled Business Processes (Business Processes belonging to different organizations);
- to identify measurable goals, which can be verified by the domain stakeholders;
- to develop quality software applications;
- to create a shared semantics, a common vocabulary, about a collaborative process.

UMM is based on the dependencies between pieces of business information and not on the exchange of documents. The goal is to understand and formalize the dependencies between collaborative processes in a given application domain.

The modelling technique consists of two parts. First of all, we must accomplish the procedures spread in the three big work areas, one for each of the three first UMM views. The last view, the *Business Service View*, is not seen as a work area but rather as a result. In the second part, we must fill in the worksheets relative to each work area. These worksheets are useful to collect all the information needed to create the UML models of the current work area.

It is important to note that the information-gathering is iterative: when we discover new pieces of information in a view, we must update the worksheets and models of the previous ones.

3.6 Case study

We will now proceed with a simple case study that will show how UMM can be used in practice. The example given here is a modified version of the one proposed by [CEN]. It does not provide all the UMM worksheets and diagrams (and is therefore a simplified use of UMM) but it features well how UMM can be used. Some of the figures and tables provided hereafter come from [CEN] but have been modified to be clearer.

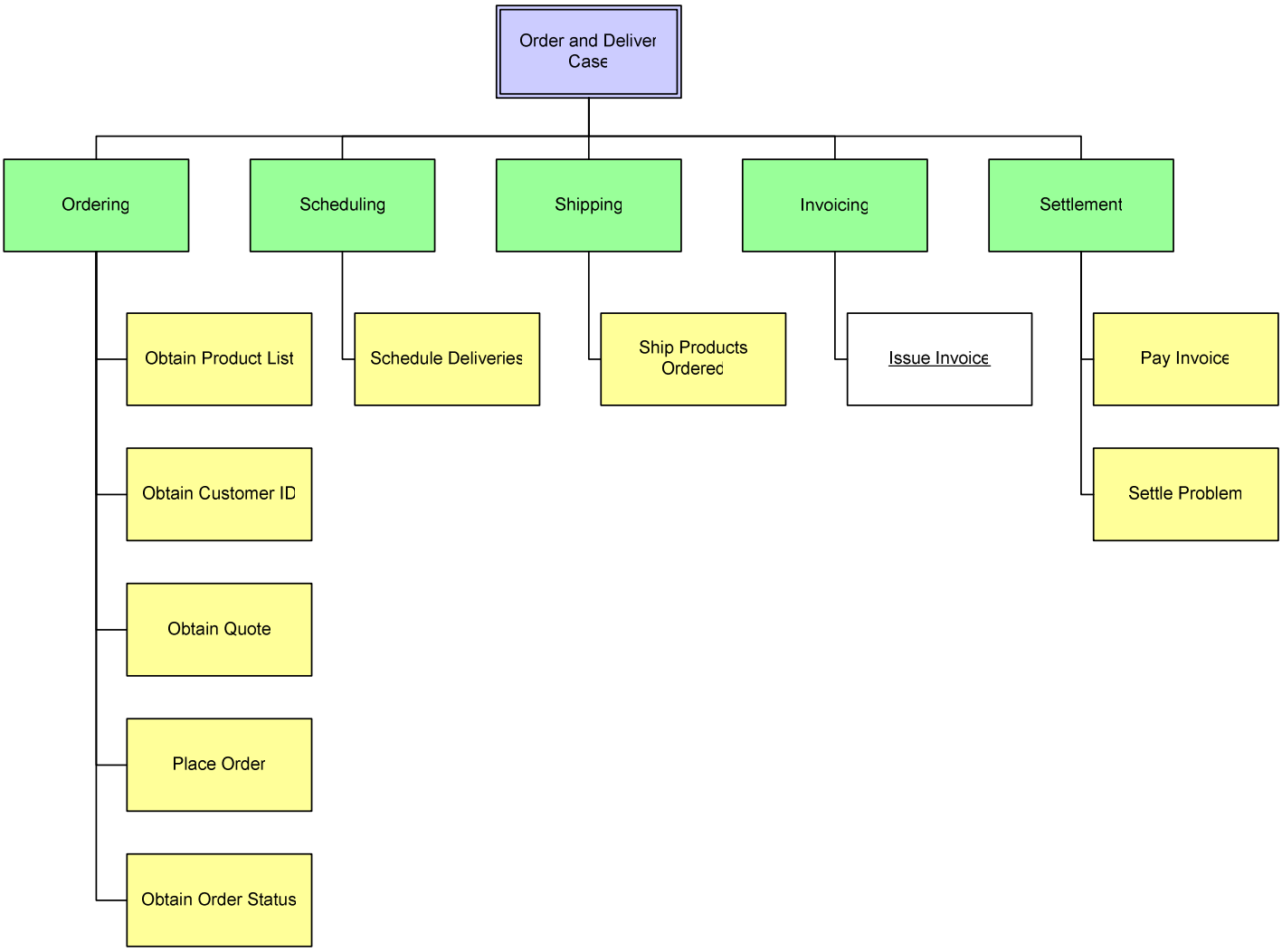
Description of the case

This example has for context a supplier that sells goods by using an electronic catalogue. The customer can check the website of the supplier to see what it is currently selling. S/He has in that way access to the catalogue and can find the entire information (description, price, etc.) relative to the products available. The customer has the opportunity to place an order directly on this website. But for doing this, s/he must be registered in the supplier's database. The registration process takes place the first time the customer orders a product. This personal information is checked and if it is valid, the system returns a Customer ID. Afterwards, the customer uses his/her Customer ID when placing an order. His/her information (solvency, etc.) is checked before any order is accepted.

The customer can consult the order status until the product is delivered. Once the order is accepted and the product is delivered, the supplier sends a shipment notice and an (electronic) invoice to the customer. The customer checks the goods received. If everything is right, s/he pays the invoice. If not, the customer contacts the supplier and the problem settlement procedure (replacement of the goods, reimbursement, etc.) begins.

Business Domain View

Figure 3-22 depicts the business domain of the studied case (this diagram could be obtained by interviewing business experts for example).



Legend:

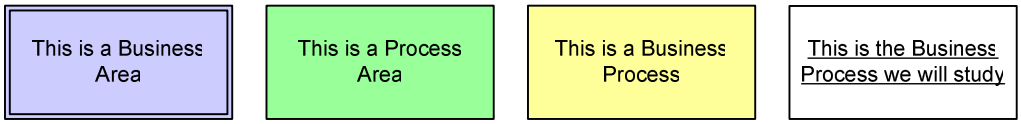


Figure 3 - 21: Business domain of the case

Description of the « Issue Invoice » process[1]

This process details the invoicing procedure between the supplier and the customer. The invoice is created by the supplier and sent to the customer, claiming payment for the goods that have been delivered under the conditions agreed by both parties. This process supports the generation of the invoice by the supplier to the customer, and covers also the treatment and the reconciliation of an inaccurate invoice.

[1] This case study will be focused on the "Issue Invoice" Business Process

The invoice can also cover the functions of debit note or credit note. The debit and credit note can be used to correct the total invoiced amount, to cancel a previous invoice or to give an additional rebate, covering the treatment and the reconciliation of an inaccurate invoice.

If the customer finds something inaccurate in the invoice, s/he can raise a dispute notice. By means of a dispute notice the customer can provide the reason for non-acceptance and can propose the corrections to be made. The supplier can use the dispute notice response to give an answer to the customer, mentioning how the inaccurate invoice will be settled. The settlement of the inaccurate invoice can be done in one of the following ways:

- A credit note is generated to cancel the previous sent invoice, together with a new invoice with the correct information;
- or a credit or debit note is sent to the customer to settle the corrections agreed between the parties.

Business Requirements View

Use case[1] diagram: Issue Invoice (*Business Process*)

This Business Process involves two partners and is thus collaborative. A collaborative process comprises at least one Business Collaboration.

The use case for the « Issue Invoice » process shows which *Business Collaboration Use Cases* are used to compose the process:

- The treatment of an invoice
- The treatment and the reconciliation of an inaccurate invoice

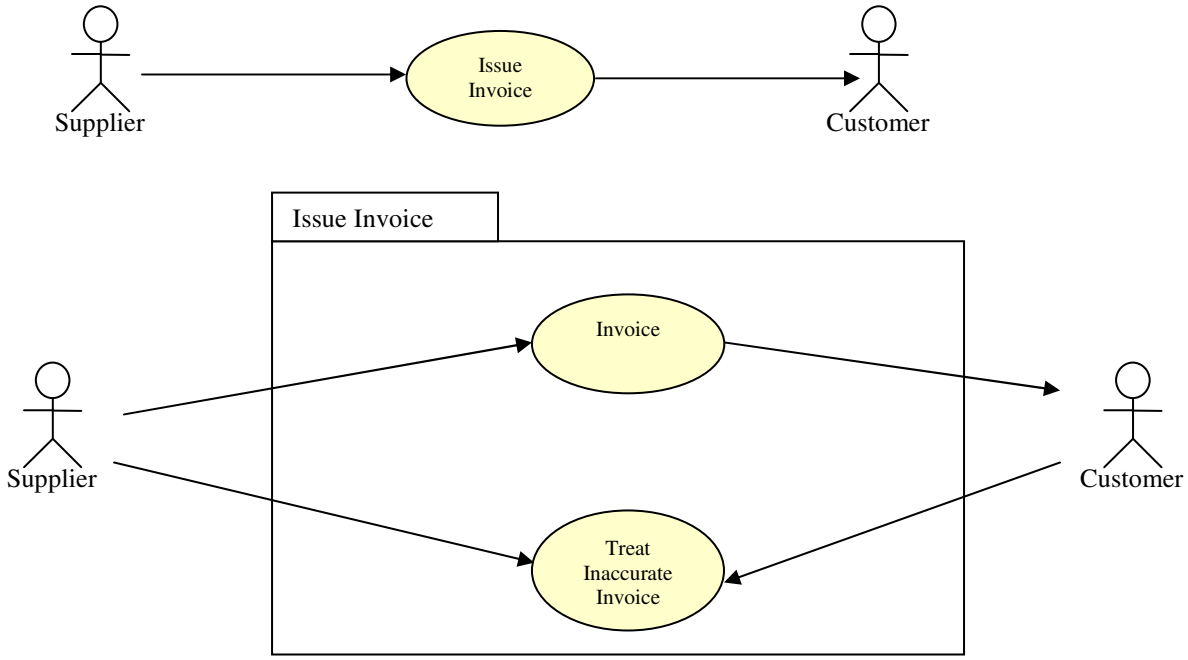


Figure 3 - 22: Use case diagram: Issue Invoice (based on [CEN])

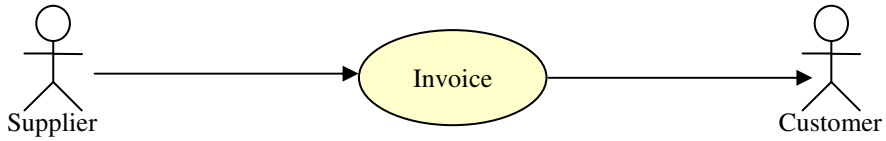
[1] As mentioned before, UMM extends UML by using the stereotype mechanism. The BDV meta-model (see Figure 3-7 above) specifies that the UMM Business Process extends the UML Use Case diagram.

For each of the Business Collaboration Use Case a description is provided.

Use case diagram: Invoice (*Business Collaboration Use Case*)

The use case for the Invoice has the following *business transaction*:

- Provide an invoice.



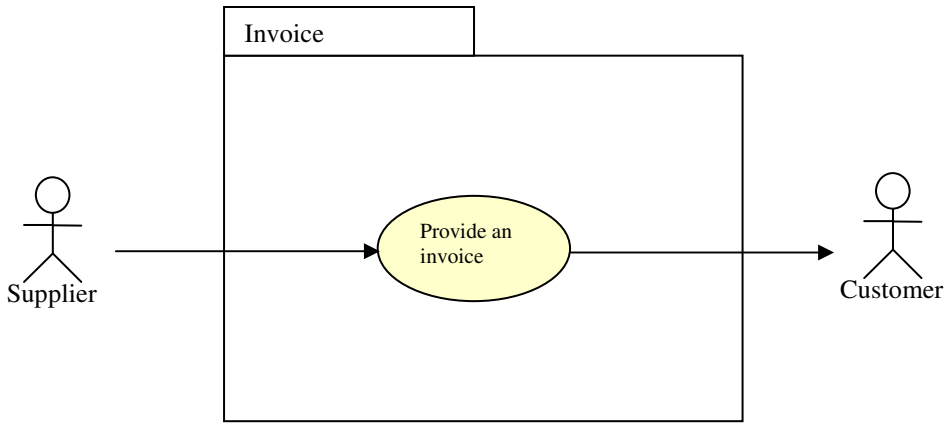


Figure 3 - 23: Use case diagram: Invoice (based on [CEN])

Use case description: Invoice (*Business Collaboration Use Case*)

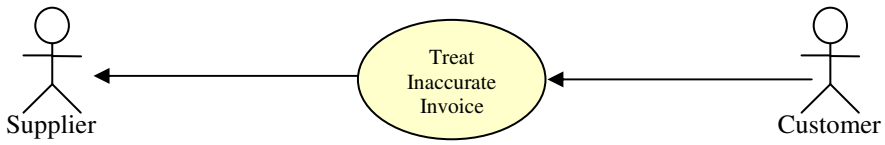
Business process name	Invoice sent by the supplier to the customer
Identifier	EEG1 e-Invoicing PT: Traditional invoice
Actors	Customer, Supplier (Optional, additional roles – Invoicee, Invoice issuer)
Description	The supplier presents to the customer for the ordered or delivered, received or consumed goods a detailed statement of trade account payable (invoice). The customer reconciles the invoice with the agreed prices and the goods or service rendered and initiates the payment remittance.
Pre-condition	Framework Agreement or Contract and order is in place with agreed prices. The supplier has provided goods according to the conditions set in the contract and, or order. The customer has received the goods.
Post-conditions	Based on the reconciled invoices the customer should issue the notification for the payments. For the inaccurate invoices the customer will generate a dispute notice to the supplier.
Scenario	Based on the agreed conditions in the contract, order and or the delivery schedule or delivery just in time, the supplier will provide goods to the customer. In function of the shipping instructions the goods will be delivered directly to the customer or to a third party warehouse or to a consignment stock (more details of the different ways are provided in the shipping cycle). Based on the agreement of the point of invoicing between the parties, the supplier will generate the invoice for the goods or services based on the goods ordered, or delivered, or received or consumed. Once the goods are delivered to the customer together with the shipment notice (despatch advice, packing list or waybill), the customer checks the invoice with the order and contract information and with the goods accepted by the customer. If there is any discrepancy found, the customer shall start the process to treat inaccurate invoices. In the other case the invoice will be submitted to the payment cycle.
Remarks	

Table 3-1: Use case description: Invoice (based on [CEN])

Use case diagram: Treat Inaccurate Invoice (*Business Collaboration Use Case*).

The use case for the Treat Inaccurate Invoice has the following *business transactions*:

- Initiate a dispute notice
- Settle the inaccurate invoice.



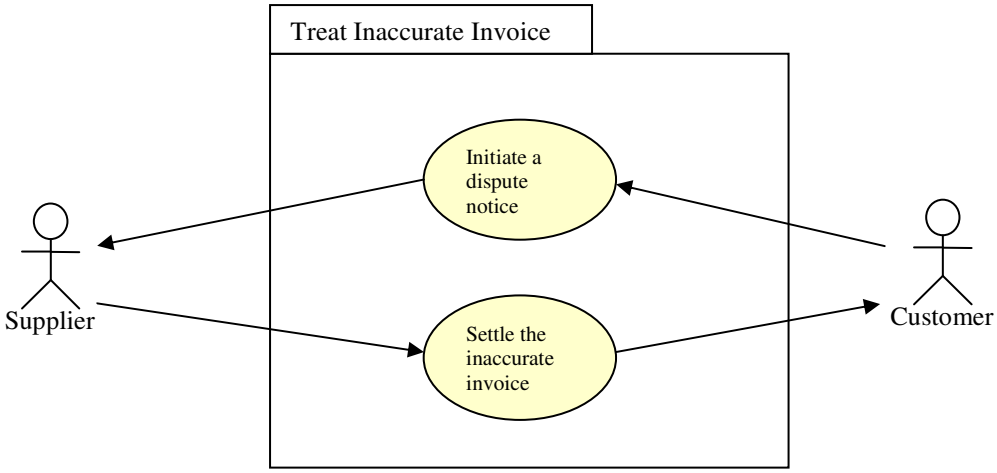


Figure 3 - 24: Use case diagram: Treat Inaccurate Invoice (based on [CEN])

Use case description: Treat Inaccurate Invoice (*Business Collaboration Use Case*)

Business process name	Treat Inaccurate Invoice
Identifier	EEG1 e-Invoicing PT: Inaccurate invoice
Actors	Customer, Supplier (Optional, additional roles - Invoicee, Invoice issuer)
Description	The Customer has found in the invoice from the supplier, a discrepancy between the invoiced goods and the received goods, or between the price conditions applied and the price conditions agreed.
Pre-condition	The customer received an inaccurate invoice from the supplier.
Post-conditions	The supplier has accepted the dispute notice raised by the customer, and the dispute is settled. The supplier rejects the dispute notice.
Scenario	Once the goods are delivered to the customer together with the shipment notice, the customer checks the received invoice with the order and contract information and with the goods accepted by the customer. If there is any discrepancy found, the customer shall generate a dispute notice for the supplier. On receipt the supplier shall check the dispute notice and shall raise a dispute notice response to inform the customer if the dispute notice is accepted or not accepted. When accepted the supplier shall inform the customer how the inaccurate invoice shall be settled. To settle the inaccurate invoice the supplier has the choice to send a credit note to cancel the previously invoice and to generate a correct invoice, or he can settle the difference by using a credit note or a debit note.
Remarks	- The credit note and the debit note are covered by the Invoice document.

Table 3-2: Use case description: Treat Inaccurate Invoice (based on [CEN])

Business Entities:

All *Business Collaborations* have a subject. This subject is a *Business Entity*. As said before, a *Business Entity* is an abstraction for any artifact that is important in the execution of a business collaboration. In the case of the « Invoice » *Business Collaboration*, the *Business Entity* is the invoice. In the case of the « Treat Inaccurate Invoice » *Business Collaboration*, the *Business Entity* is the dispute notice.

We could describe all the attributes and states of these *Business Entities* but it was not done in [CEN] and we cannot find out the exact behaviour of the *Business Entities*. However, the invoice document is described in [CEN] and we will talk about it when we will study the *Business Information* in the *Business Transaction View* of this case study.

Business Transaction View

For each of the Business collaboration use cases mentioned earlier, the corresponding activity diagrams are presented.

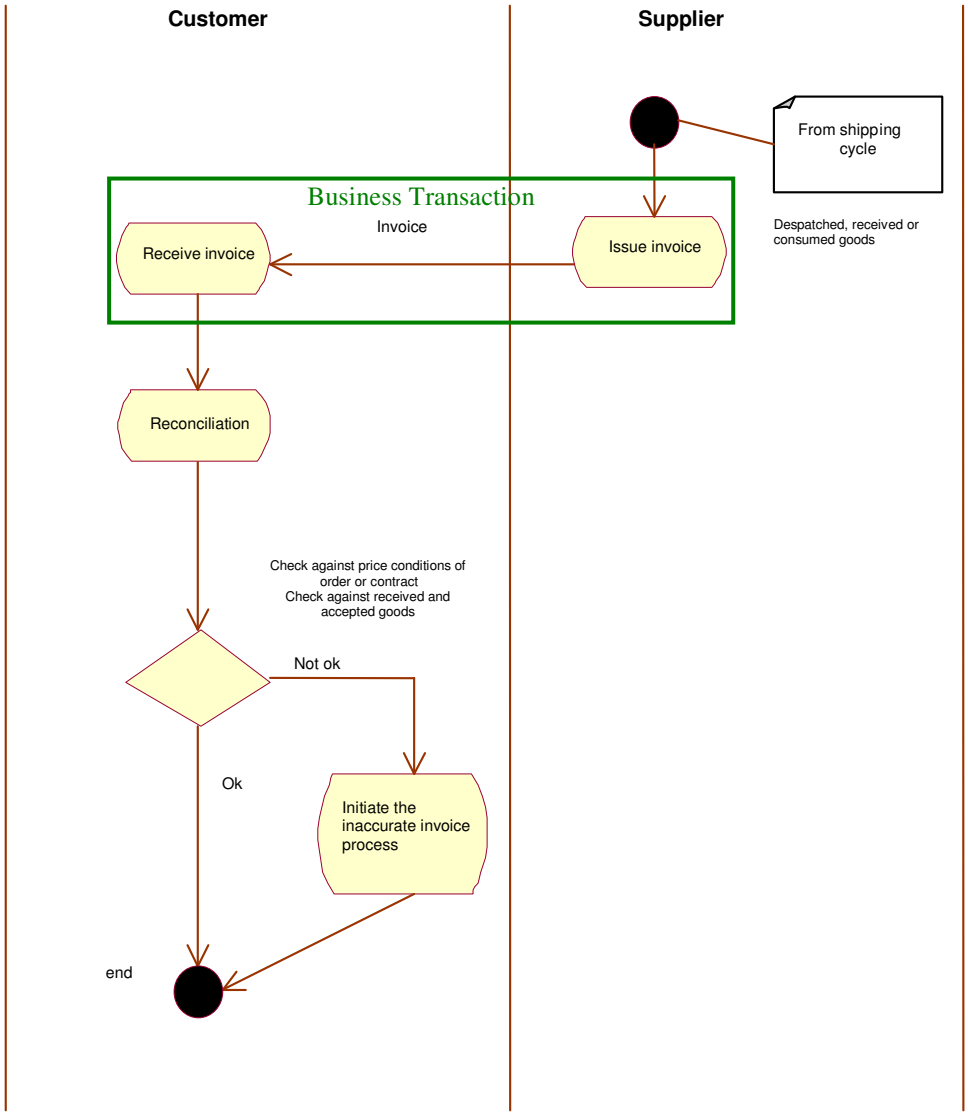


Figure 3 - 25: Activity diagram: Invoice (based on [CEN])

Activity diagram description: Invoice (Business Collaboration Use Case)

Based on the conditions agreed between the customer and supplier, the supplier will initiate the invoicing of goods delivered. The invoice is based on the despatched goods. The invoice is created after the shipment of the goods and is based on the goods present in the consignment. This is the normal case for direct delivery of goods from the supplier to the customer.

The supplier sends the invoice to the customer. When the customer receives the invoice, s/he checks that the price conditions applied against the price conditions agreed and specified in the contract or order. The customer checks also the goods invoiced against the goods received. If no discrepancies are detected, the invoice is accepted and will be paid. If there is any discrepancy detected by the customer, the customer should initiate the inaccurate invoice procedure to advise the supplier by a non-conformity notification and corrective action has to take place.

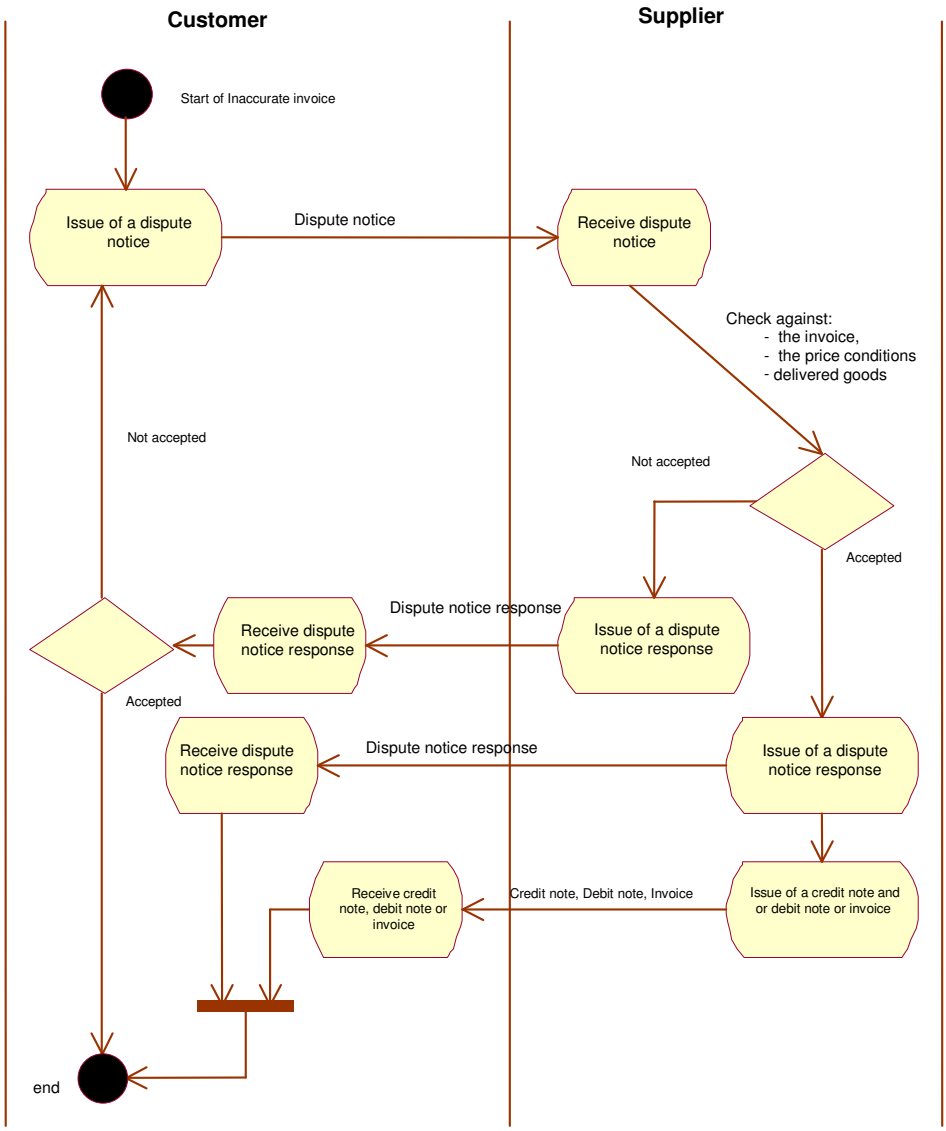


Figure 3 - 26: Activity diagram: Treat Inaccurate Invoice (from [CEN])

Unfortunately, this diagram does not say what happens if the credit or debit note is refused by the customer. [CEN] says nothing about this possibility.

Activity diagram description: Treat Inaccurate Invoice (*Business Collaboration Use Case*)

If there is any discrepancy detected in the invoice, the customer initiates the inaccurate invoice procedure by raising a dispute notice. By this dispute notice s/he informs the supplier about the discrepancies found and can propose the relevant corrective action to be taken by the supplier. When the supplier receives the dispute notice, he checks the non-conformity notification with his/her information concerning the invoice, the price conditions and the delivered goods.

Based on his/her finding, the supplier will accept or not accept the dispute notice and the corrective action proposed. If the dispute notice is not accepted, the supplier will raise a dispute notice response to the customer, mentioning the reason of non-acceptance and/or propose another corrective action. On receipt of the dispute notice response, the customer will, after evaluation of the response, raise a new dispute notice containing a new proposal or confirm the acceptance of the proposal made by the supplier in the dispute notice response.

If the dispute notice is accepted, the supplier will raise a dispute notice response to inform the customer and will take the corrective actions. To correct an inaccurate invoice, the supplier has the option to cancel the inaccurate invoice by using a credit note and to generate the correct invoice, or he can raise a credit note or debit note for the difference between the amount of the inaccurate invoice and correct amount. On receipt of the dispute notice and the credit note, debit note and/or invoice, the customer will check them against the dispute notice initiated by him.

The business collaboration specifies for each of the activity diagrams mentioned above the input and output triggers, constraints and system boundaries for the business transactions, business collaboration protocols and their interrelationships.

Business Collaboration description: Invoice (*Business Collaboration*)

Business Collaboration	
Identifier	EEG1 e-Invoicing PT: traditional invoice
Description	The supplier raises and sends an invoice to the customer. When the customer receives the invoice, he checks the invoice against the price conditions and the invoiced goods against the received and accepted goods. If the invoice is correct, the invoice is accepted and submitted to the payment administration. If there is any discrepancy detected the customer shall initiate the inaccurate invoice process.
Partner Types	Customer Supplier
Authorized Roles	Customer (Customer's agent, Buyer or Buyer's agent, Invoicee) Supplier (Supplier's agent, Seller or Seller's agent, Invoice issuer)
Legal Steps/Requirements	None
Economic Consequences	None

Initial/Terminal Events	• Initial: the supplier sends the invoice • Terminal: the customer accepts the invoice or initiate the inaccurate invoice process
Scope	To request payment for the ordered and delivered goods.
Boundary	Not defined yet
Constraints	The supplier shall have full traceability of his invoice to make sure it has been received. Failing this technical acknowledgement, the supplier shall re-issue his/her invoice message

Table 3-3: Business Collaboration description: Invoice (based on [CEN])

In a business collaboration agreement between the customer and the supplier, the way of using the Invoice can be described. The following transactions concerning the Invoice can be specified as:

1. The supplier generates and sends an Invoice, the customer has to give an acknowledgement of receipt of the Invoice.
2. The supplier generates and sends an Invoice. No acknowledgement message is used between the parties to confirm the reception of the invoice message.

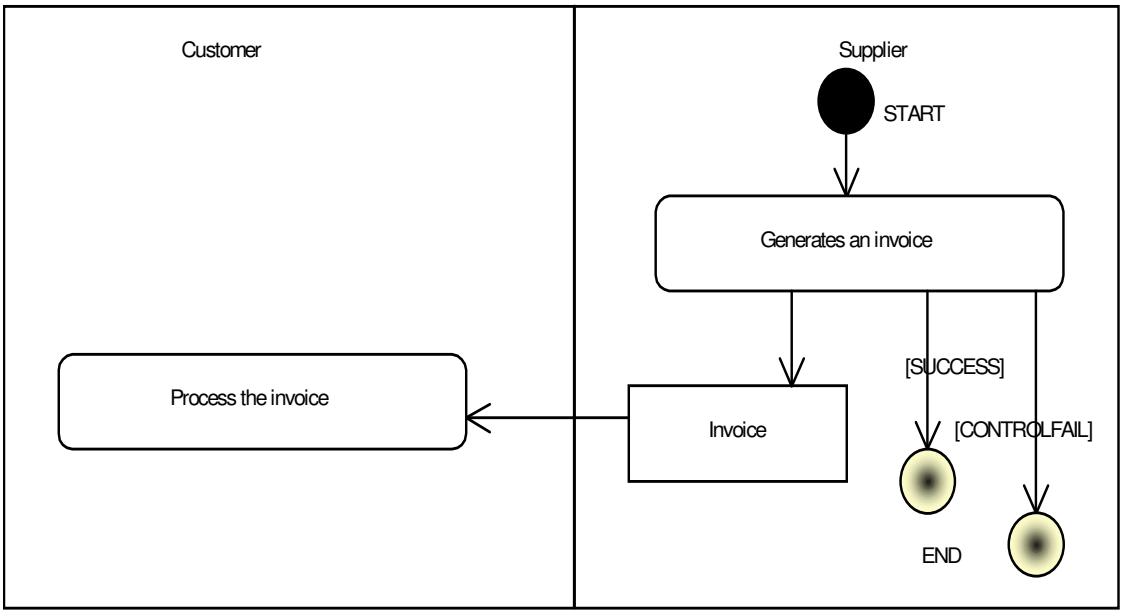


Figure 3 - 27: Business Transaction activity graph: Invoice (from [CEN])

Form: Business Transaction Transition Table					
From Id	: EEG1 e-Invoicing PT: Invoice				
From Activity	**From Role**	**Document**	**To Activity**	**To Role**	**Guard Condition**
START	N/A	N/A	Generate an invoice	Supplier	NONE
Generate an invoice	Supplier	Invoice	Process the invoice	Customer	NONE
Generate an invoice	Supplier	N/A	SUCCESS	N/A	Invoice received
Generate an invoice	Supplier	N/A	CONTROLFAIL	N/A	Invoice not received

Table 3-4: Business Transaction transition table: Invoice (from [CEN])

Unfortunately, it seems that Table 3-4 does not correspond exactly to the diagram depicted by Figure 3-28. [CEN] does not give further explanations about this. As mentioned before, this case study, proposed in [CEN], is a simplified use of UMM and some things are incompletely described.

Business collaboration description: Treat Inaccurate Invoice (Business Collaboration)

Business Collaboration	
Identifier	EEG1 e-Invoicing PT: Inaccurate invoice
Description	The customer raises and sends a dispute notice to the supplier to report any discrepancy in an invoice. When the supplier receives the dispute notice, he checks the referred invoice with the price conditions and the invoiced goods with the delivered goods. Based on his finding, he raises and sends a dispute notice response to reject the dispute notice or to propose another settlement, or he raises and sends a dispute notice response together with a credit note, debit note and or invoice to settle the inaccurate invoice.
Partner Types	Customer Supplier
Authorized Roles	Customer (Customer's agent, Buyer or Buyer's agent, Invoicee) Supplier (Supplier's agent, Seller or Seller's agent, Invoice issuer)
Legal Steps/Requirements	None
Economic Consequences	None
Initial/Terminal Events	• Initial: the customer sends a dispute notice • Terminal: the inaccurate invoice is settled, or the supplier

	propose another settlement
Scope	To settle any discrepancy in the invoice.
Boundary	Not defined yet
Constraints	The customer shall have full traceability of his dispute notice to make sure it has been received. Failing this technical acknowledgement, the customer shall re-issue his dispute notice

Table 3-5: Business Collaboration description: Treat Inaccurate Invoice (based on [CEN])

In a business collaboration agreement between the customer and the supplier, the way of using the inaccurate invoice process initiated by the customer can be described. The following transactions concerning the inaccurate invoice can be specified as:

1. The customer generates a dispute notice to inform the supplier of any discrepancy in the invoice, the supplier has to give an acknowledgement of receipt of the dispute notice. The supplier sends a dispute notice response and if valid a credit note, debit note and or invoice to settle the inaccurate invoice. The customer acknowledges the receipt of the Dispute notice response to the supplier.

2. The customer generates a dispute notice to inform the supplier of any discrepancy in the invoice, and the supplier responds directly with a dispute notice response. No acknowledgement messages are used between the parties.

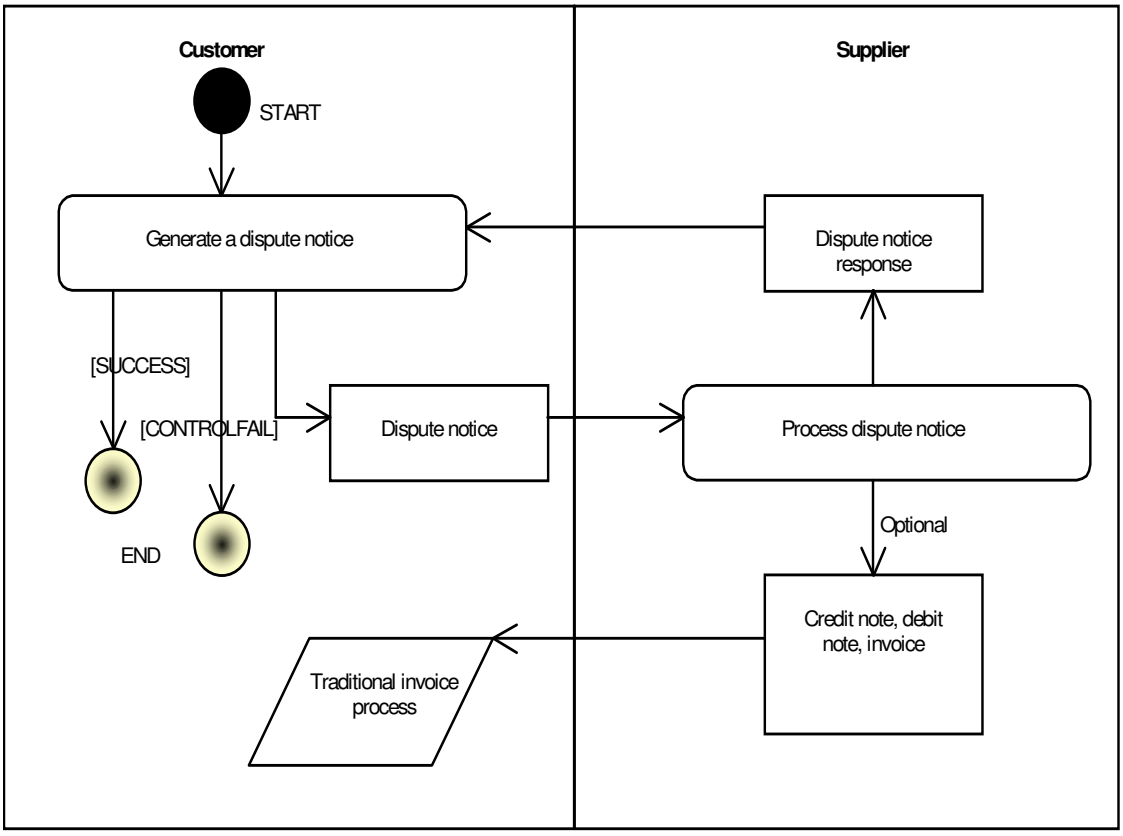


Figure 3 - 28: Business Transaction activity graph: Treat Inaccurate Invoice (from [CEN])

Form: Business Transaction Transition Table					
From Id	EEG1 e-Invoicing PT: Inaccurate invoice				
From Activity	**From Role**	**Document**	**To Activity**	**To Role**	**Guard Condition**
START	N/A	N/A	Generate a dispute notice	Customer	NONE
Generate a dispute notice	Customer	Dispute notice	Process dispute notice	Supplier	NONE
Process dispute process	Supplier	Dispute notice response	Generate dispute notice	Customer	NONE
Process dispute process	Supplier	Credit note, debit note and or invoice	Traditional invoice process	Customer	NONE
Generate dispute notice	Customer	N/A	SUCCESS	N/A	Dispute notice accepted
Generate dispute notice	Customer	N/A	CONTROLFAIL	N/A	Dispute notice not received

Table 3-6: Business Transaction transition table: Treat Inaccurate Invoice (from [CEN]).

The goal of the *Business Transactions* is to identify the individual transactions that implement the operations of business collaboration. A transaction is made up of several activities and each activity has an authorized role in order to initiate that activity.

Business transactions and authorized roles: Invoice (*Business Collaboration*)

Business Transaction	
Description	The Invoice is used for claiming payment for goods or services supplied under conditions agreed between the supplier and the customer.
Pattern	Notification
Business activities and associated authorized roles	See EEG1 e-Invoicing PT: Traditional invoice
Constraints	Not defined yet
Initiating/Requesting Partner Type	Supplier
Initiating/Requesting Activity Role	Supplier (Supplier's agent, Seller or Seller's agent, Invoice issuer)

Initiating/Requesting Activity Document	Invoice

Table 3-7: Authorized roles: Invoice (from [CEN])

Business transactions and authorized roles: Treat Inaccurate Invoice (*Business Collaboration*)

Business Transaction	
Description	The Dispute notice is used by the customer to inform the supplier of any discrepancy detected in an invoice, and to correct the inaccurate invoice.
Pattern	Notification/Response
Business activities and associated authorized roles	See EEG1 e-Invoicing PT: Inaccurate invoice
Constraints	Not defined yet
Initiating/Requesting Partner Type	Customer
Initiating/Requesting Activity Role	Customer (Customer's agent, Buyer or Buyer's agent, Invoicee)
Initiating/Requesting Activity Document	Dispute notice
Responding Partner Type	Supplier
Responding Activity Role	Supplier (Supplier's agent, Seller or Seller's agent, Invoice issuer)
Responding Activity Document	Dispute notice response
Other responding Activity Document	If the dispute notice is accepted by the supplier, the supplier raises and sends a credit note, debit note and or invoice to settle the discrepancy between the inaccurate invoice and the correct invoice. On receipt of these documents the customer shall initiate the reconciliation of the documents in respect of the dispute notice.

Table 3-8: Authorized roles: Treat Inaccurate Invoice (from [CEN])

Business Information:

[CEN] uses the following definition of the invoice document:

In trade, the invoice message is used to request the payment for the goods that have been ordered or received or consumed. Usually, the supplier invoices the customer when the goods are delivered or the services provided. In this case the invoice can be created at the moment of despatch or when the customer or a third party gives the acknowledgement that the goods are received.

When there are discrepancies between the despatch advice, the invoice and the goods actually received or rejection of goods for quality reasons, the customer will claim a credit note from the supplier before paying the invoice. A credit note or debit note may be issued in the case of retrospective price change.

The Invoice message, developed by eBES/EEG1 Project team e-Invoicing is intended to cover the following functions:

- to invoice the goods based on one delivery
- to generate a pro-forma invoice for customs purposes
- to generate a pre-invoice before the delivery of the goods
- to have a consolidated invoice covering several deliveries done in a predefined time frame
- to generate a credit note to cancel a previous sent invoice
- to generate all kind of credit notes, for example to settle claims for damaged goods, wrong deliveries, invoice errors, price changes, etc.

We will now provide the diagrams showing the structure of the invoice document (created by customizing and combining Business Objects stored in a library). We do not describe all the attributes present in these diagrams because it is not important in our discussion. A complete description of these attributes can be found in [CEN].

Figure 3 - 29: Invoice Document (from [CEN])

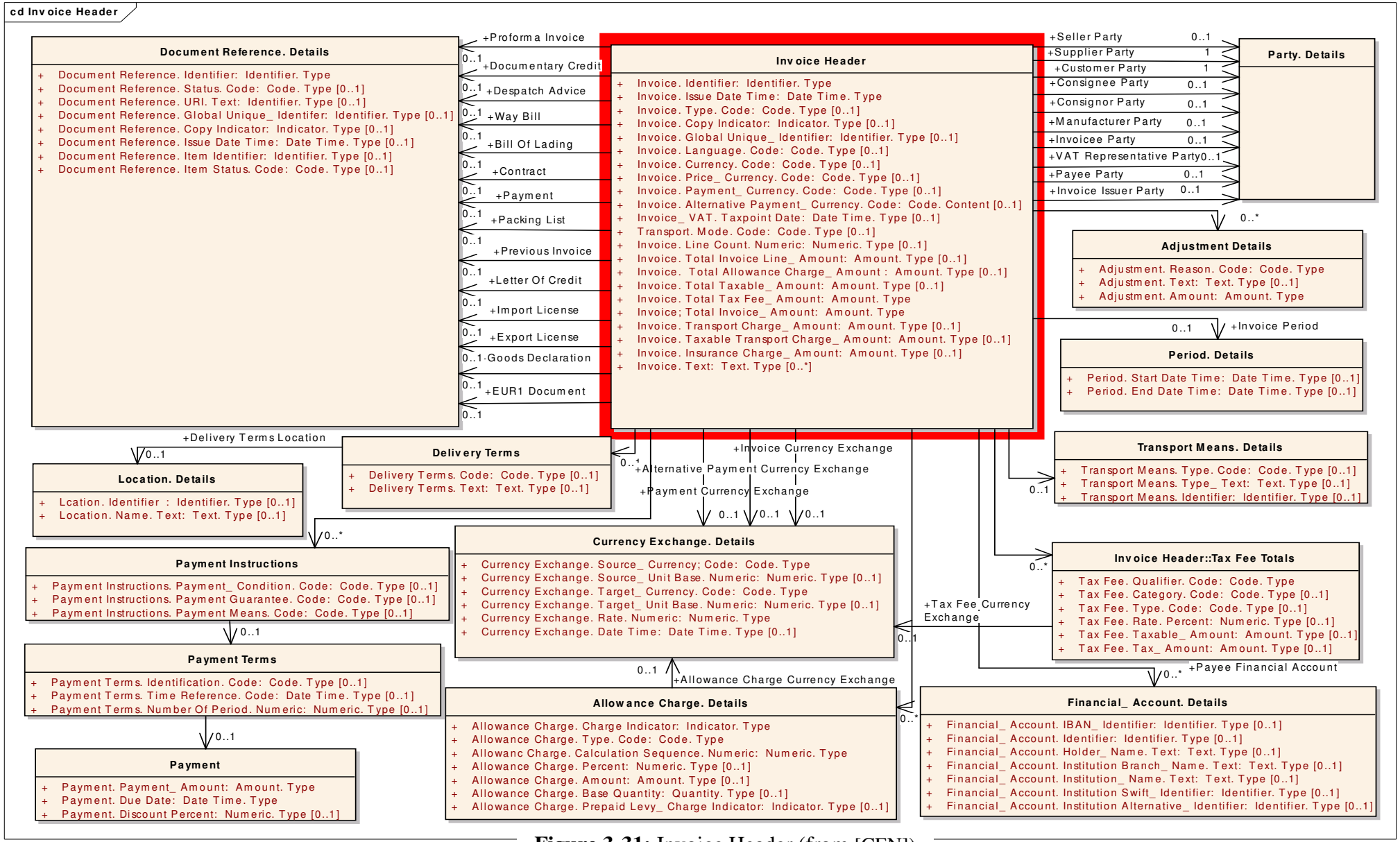


Figure 3-31: Invoice Header (from [CEN])

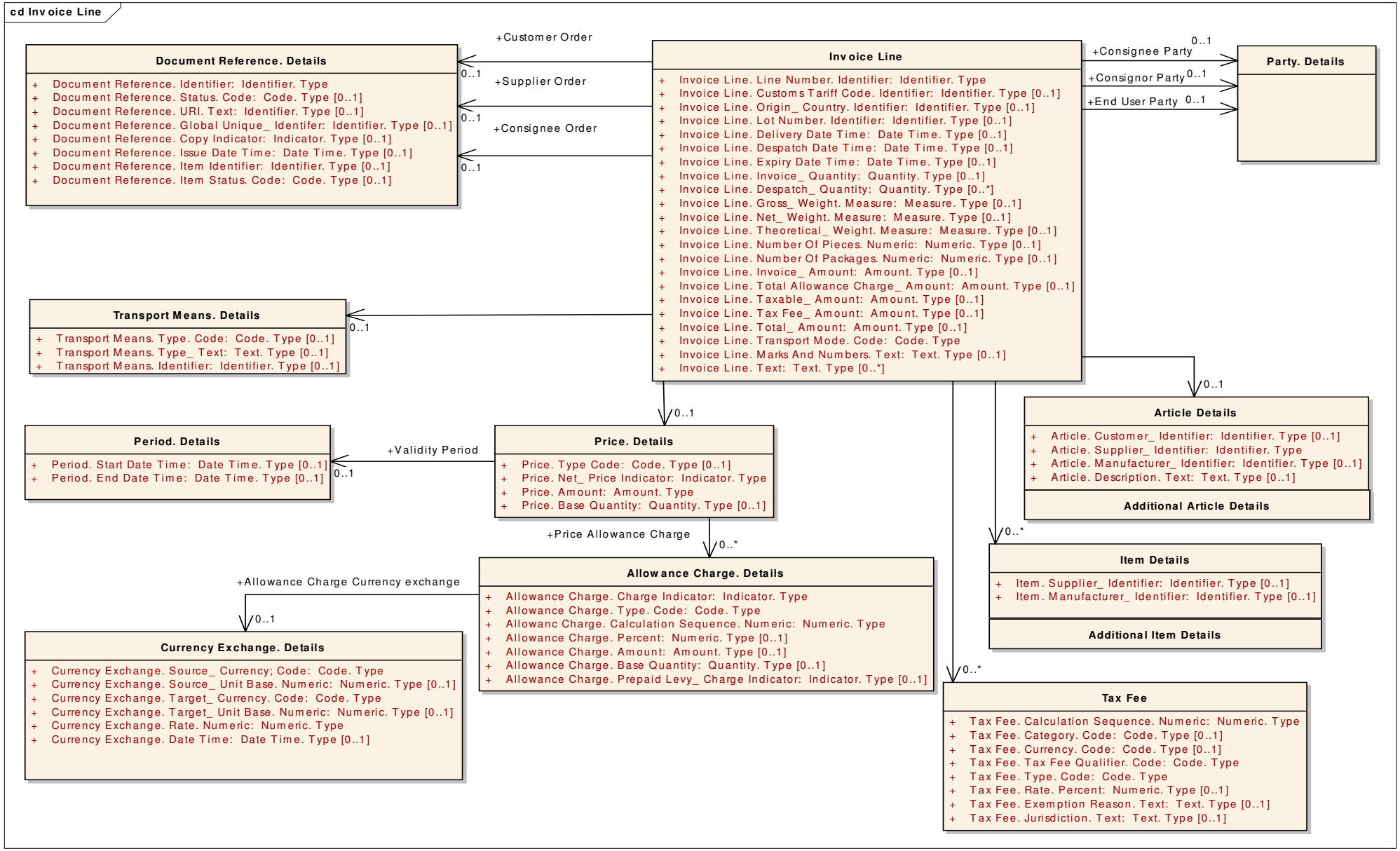


Figure 3-32: Invoice Line (from [CEN])

3.7 Review of UMM.

This section presents our critical opinion of the UMM. We begin with the negative points before showing the positive aspects of UMM.

3.7.1. UMM problematic issues

First of all, the documentation of UMM is not completely precise. The vocabulary used in the User Guide ([UMM UG]), which can be considered as the most elaborated documentation on UMM, is very ambiguous. Here are some examples: Collaboration Activity, Business Collaboration, Activity; Business Transaction Activity, Business Transaction, transaction (in the classic economic sense); Process (in the UMM sense), process (in the classic sense); Activity (in the UMM sense), activity (in the classic sense); sub-Business Process, Activity; Business Area, Process Area, Business Process Area. The authors could have chosen other less close, more evocative terms. It can sometimes be really difficult to understand what we are talking about in certain sentences. It is one of the major reasons why the User Guide, taken alone, is insufficient to understand UMM. Other documents from different sources are required to really understand certain parts of it. A lot of concepts are poorly or only partially explained. Multiple readings of the User Guide are necessary to catch the meaning of some concepts or certain relationships between concepts. Unfortunately, understanding certain concepts can sometimes be compared to collecting pieces of a puzzle and trying to put them in the right order. For all these reasons, it is our opinion that the User Guide misses its objective of introducing UMM correctly.

Secondly, the relationship between *Business Process* and *Business Collaboration* is not clear. This is really problematic because these concepts are the basis of UMM. Actually, the UMM meta-models, which we obviously consider as the source of reference, says that a *Business Process* is divided into one or more several *Business Collaborations*. However, other sources, including [UMM UG] and [Hofreiter&al. 2004] say that a *Business Collaboration* is a category of *Business Process* (a *Business Collaboration* is a collaborative *Business Process*). This inconsistency is the worst problem we have identified in the documentation about UMM.

Thirdly, the worksheet approach, while relevant and useful, is very heavy. There are quite many redundancies in some worksheets and the separation between steps in a same worksheet is not always obvious. A major problem is the sometimes very long description of states, transitions, transition-triggers and transition conditions. Their understanding requires a lot of efforts. For example, footnoted UML statecharts could be used to make these descriptions more intuitive and synthetic. As a general criticism, a lot of worksheets are used but only a few diagrams are provided. The goal of the worksheets should be limited to the collection of the needed information to create the diagrams.

Fourthly, the User Guide says that libraries contain descriptions that can be reused and customized but doesn't say if there is a real consensus about these. Their effective use to model critical parts of business collaborations is questionable for two reasons:

- business engineers will not necessarily trust these descriptions. They may not agree with the chosen modelling approach;
- it is a good idea to try to standardise certain things to avoid reinventing the wheel each time but we must not forget that enterprises evolve in a world of high competition. Each enterprise wants to gain an advantage on its competitors and this cannot be done if everybody uses the same descriptions. Each company would prefer to optimize its processes and keep its methods secret to be able to propose the best solution to the customer.

Fifthly, there is confusion between top-down and bottom-up modelling in UMM. The decomposition of the *Business Domain* into *Business Processes*, *Business Collaborations*, *Business Transactions*, etc. is clearly top-down. On the other hand, the reuse, customization and combination of

existing descriptions are obviously bottom-up oriented. It is difficult to put a clear border between the top-down phase and the bottom-up phase. That should be made more explicit.

3.7.2. UMM strengths

Firstly, the top-down decomposition is a good idea. It helps to represent large Business Domains and in doing so, to reduce the complexity.

Secondly, the approach taken consisting in creating "bridges" between domain experts, business analysts and software developers is excellent. The system of views is a really good idea. Each of them is dedicated to a particular kind of professionals and depicts a different granularity level.

Thirdly, the libraries of reusable descriptions are a very good method, if a sufficient consensus is reached, to create a common language for a sector of activity and to facilitate business collaborations in that sector. It has the big advantage of making interoperability of e-business systems possible. It is also very useful for reducing the amount of time, and therefore the financial investment, needed to model a domain and to implement the collaborations with e-business technology.

Fourthly, the idea of focussing on the economic commitments (by reusing REA) is clearly interesting for the enterprises. The trading partners are not really interested in modelling languages and in information systems as such. What they are interested in is the added value that these can bring to their business. Concretely, UMM, combined with an e-business platform, brings them a faster way of discovering partners, negotiating and making business with them.

3.8 Collaborative concepts that could be added to UEML

In this chapter, we have presented the methodology and the concepts that underlie UMM. As we have seen, this modelling language is rather complicated and comprises many concepts. Some of these concepts are specific to UMM (the elements extending UML for instance) and provide details that are not useful for allowing e-business modelling with UEML. Therefore, only the most essential[1] concepts, in terms of trading collaboration, will be adapted to UEML. In what follows, we will discuss the concepts that should be kept.

Obviously, the REA part must be totally reused in UEML because it defines what a real world trading collaboration is. Therefore, the following concepts should be kept without changes: *Partner type*, *Economic Contract*, *Economic Commitment*, *Economic Event*, *Economic Resource*, and *Economic Resource Type*.

The concepts of *Business Collaboration* and *Business Transaction* are relevant to model e-business relationships. Indeed, e-business implies collaboration, and transactions represent the steps that must be performed in order to accomplish a business collaboration. However, these concepts are quite complex in UMM and should be adapted to (i.e. simplified in) UEML.

The concept of *Business Information* is well suited to model information exchanges between the trading partners. Obviously, e-business transactions imply business information exchanges. This concept should therefore be adapted to UEML.

The other UMM concepts will not be retained because they are not strictly necessary for the level[2] we want to reach in e-business modelling with UEML.

[1] The set of concepts to be added to UEML is intended to be minimal.
[2] A high-level (rather conceptual, i.e. not close to software implementation) is intended.

4. Adding collaborative concepts to UEML: EBCML

4.1 Introduction

In the very beginning of this work, it was planned to study both UEML and UMM and then integrate UMM concepts into UEML by applying the "strategy for UEML" (which is described in chapter 2). But it has quickly become clear that it was not really possible to compare UEML and UMM because of their different goals.

UEML models the internal[1] processes of an enterprise while UMM models the business collaborations between several enterprises. UMM is not a "classical" enterprise modelling language and cannot be compared to IEM, EEML or Grai. It was therefore not possible to apply "the strategy for UEML" to UMM and UEML. The search for common concepts would be nearly impossible because UMM and UEML are usually used to represent very different things.

However, it is possible to add concepts coming from UMM to UEML, but without using the "strategy for UEML". The resulting modelling language, which is a new version of UEML, would be able to represent at the same time the internal processes of an enterprise *and* the business collaborations in which this enterprise is involved. UEML 1.0 focuses on the workflows between intra[1]-enterprise activities. But there are *activities* that need *resources* coming from outside the enterprise, and these resources must be bought. On the other hand, there are activities that produce *resources* that must be sold to customers. Therefore an enterprise cannot survive without collaborating with partners. Indeed, an enterprise is a kind of black box that takes several things in input and transforms these inputs in order to produce things in output. Obviously, this process is supposed to create an added value, which is the raison d'etre of the enterprise.

The new version of UEML presented in this chapter is called EBCML, which stands for "Enterprise and Business Collaboration Modelling Language". EBCML could be used as an input in the development of e-business information systems. Nowadays, enterprise integration, flexibility and interoperability can be achieved in an efficient way through the use of e-business platforms. A good model of the enterprise domain is needed to create such software systems. The proposed language would model the business domain in a way that is technology-independent and the resulting models could be implemented in any suited e-business technology (an ebXML infrastructure for instance).

EBCML comprises many concepts presented in chapters 2 and 3. Naturally, all UEML concepts are retained in EBCML. Collaborative concepts, derived from certain UMM elements, are also part of the language.

EBCML is described in the following. In section 4.2 and 4.3, its meta-model is presented. Then a description of its modelling elements and its graphical concrete syntax are provided (sections 4.4 and 4.5). Finally, its use is demonstrated in a case study in section 4.6.

4.2 EBCML meta-model

EBCML contains all the concepts we have retained from UMM and UEML. Since we have kept all UEML elements, the EBCML meta-model is based on the UEML meta-model. The difficulty was to add the UMM concepts in an appropriate way. Since UEML and UMM have different goals, it was not easy to create a "bridge" between the two worlds. The link was made possible through the introduction of the *Business Interaction*, which is really a keystone concept in EBCML.

[1] Even if UEML 1.0 could be used for modelling processes spanning over several organisations, it lacks constructs (concepts such as event, interaction, etc.) to model trading collaborations.

This new concept has a meaning in both enterprise modelling and collaboration modelling. In the UEML world, the Business Interaction is an entity representing the flows between the enterprise and its environment (i.e. other enterprises). It clearly shows where collaborations are needed in the operational processes of the enterprise. In the UMM world, a *Business Interaction* is a step inside a business collaboration and represents an exchange between two enterprises. Each of these enterprises plays a role, defined contractually, in the collaboration.

The following two figures show the EBCML meta-model. Figure 4-1 shows the relationships among concepts. As in UMM, the concepts are grouped in modelling views. Each concept is member of at least one view. The points of colour represent the belonging of a concept to a particular view. The different views will be described later in this chapter.

Figure 4-2 presents the attributes of all concepts. These attributes are sometimes very general. This choice was made in order to give enough freedom to the modeller and let him/her customize the attributes in function of his/her needs.

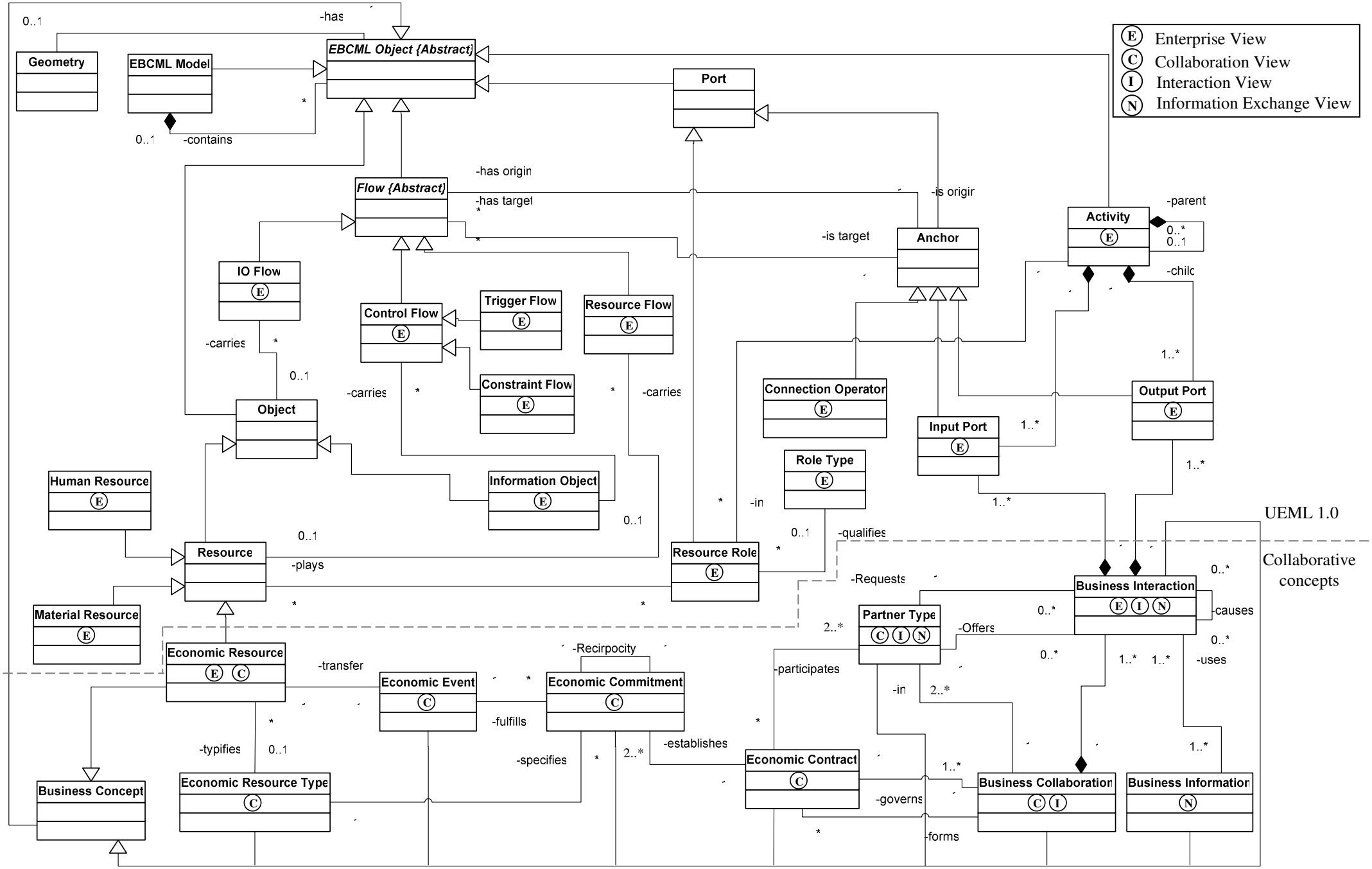


Figure 4-1: EBCML meta-model: relationships

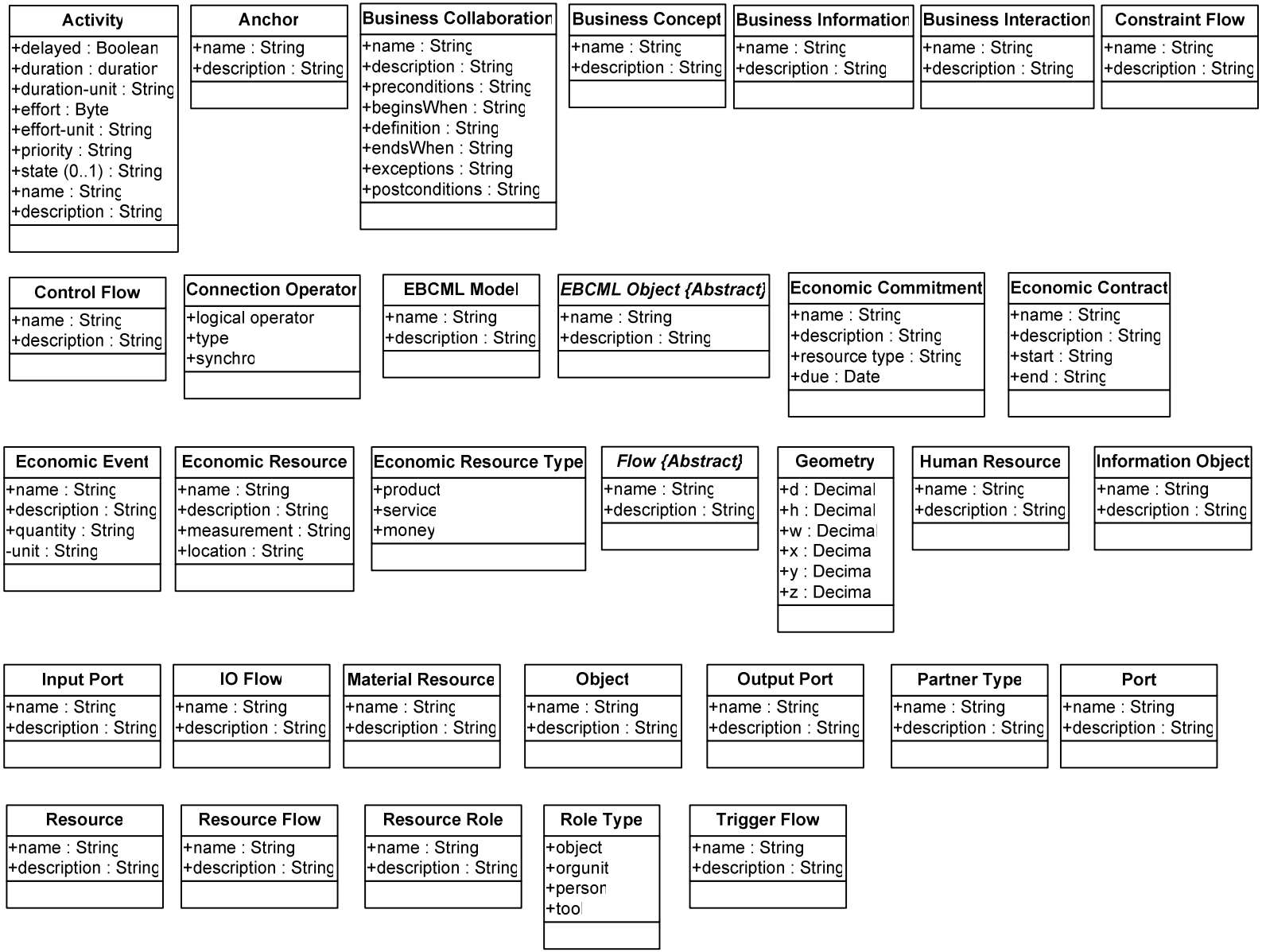


Figure 4-2: EBCML meta-model: attributes

4.3. Relationships between EBCML concepts

All the EBCML concepts come either from UMM or from UEML. These concepts have already been described in the previous chapters. Therefore, it is useless to define them once more here. We will rather explain how the concepts are linked and how we create a model in EBCML. Words in italics refer to EBCML concepts.

First of all, we place the *Activities*. An *Activity* represents a unit of work in the modelled enterprise. The execution of an *activity* may need the use of *Resources*. So, *Resources* may play a role (a *Resource Role*) in the progress of an *Activity*. A *Resource Role* can be of several types (*Role Types*): tool (a machine for instance), person (performing a job or handling a machine) or organisation unit (a department of the enterprise for example).

Each *Activity* has at least one *Input Por*t and one *Output Port*. These ports allow the *activity* to communicate with its environment. Indeed, *Flows* can connect *Activities* together. *Flows* define the choreography of the *Activities* and may carry *Objects* between *Activities*. These *Objects* can be *Resources* or *Information Objects*. A *Resource* is either a *Human Resource*, a *Material Resource* or an *Economic Resource* (an *Economic Resource* is something that is exchanged during a *Business Collaboration*). As mentioned before, a *Resource* may play a *Resource Role* (of a certain *Role Type*) in the execution of an *Activity*.

Connection Operators can be used to decompose a *Flow* into several *Flows* or to merge several *Flows* into a single one. The *Connection Operators* can represent logical relationships such as AND, OR, XOR.

Until now, we have described the concepts used to model the workflows between intra-enterprise activities. However, as said earlier in this chapter, there are *Activities* that need *Resources* coming from other enterprises or that produce *Resources* that are useful to other enterprises. The *Business Interaction* concept has been created for this reason. A *Business Interaction* involves two *Partner Types* (a provider and a customer generally). This modelling element has at least one *Input Port* and one *Output Port*. The *Resources* going to or coming from other enterprises are carried on *Flows* that connect an *Activity* and a *Business Interaction*. In order to accomplish a *Business Interaction*, it is necessary to exchange *Business Information*. A *Business Interaction* is performed in order to carry out a *Business Collaboration*.

A *Business Collaboration* is made of several *Business Interactions* that must be performed in a certain order. At least two *Partner Types* must participate in a *Business Collaboration*. A *Business Collaboration* is described by an *Economic Contract*. The *Economic Contract* establishes the *Economic Commitments* of the participators. Each *Partner Type* commits itself to trigger an *Economic Event*. An *Economic Event* consists in transferring an *Economic Resource* to a partner. There are several *Economic Resource Types*: product, service or money. Since we are talking about commercial practices, the *Economic Commitments* and *Economic Events* always go by two: one of the partners provides a product or a service and the other partner gives money in exchange.

The case study at the end of this chapter will show a concrete example of the use of EBCML.

4.4 The views

The EBCML comprises many concepts and some of them represent very different things. For this reason, it is not desirable to put all the modelling elements in a unique diagram. It is more relevant to group the concepts that are directly related and put them in a dedicated view. Each view describes a particular aspect of the domain modelled. We have identified four groups of related concepts in the meta-model and each of these groups has its own modelling view. The four views are the *Enterprise*

View, the *Collaboration View*, the *Interaction View*, and the *Information Exchange View*. Each view is shortly described hereafter.

4.4.1 Enterprise View

The *Enterprise View* represents the intra-enterprise structure and workflows. It also shows the relationships, which are flows, between internal activities and e-business interactions. This view comprises all the major UEML modelling elements. The only true new concept is the Business Interaction element.

The *Enterprise View* consists of the following modelling elements:
- *Activity*
- *Business Interaction*
- *Constraint Flow*
- *Control Flow*
- *Connection Operator*
- *Economic Resource*
- *Human Resource*
- *Information Object*
- *Input Port*
- *IO Flow*
- *Material Resource*
- *Output Port*
- *Resource Flow*
- *Resource Role*
- *Role Type*
- *Trigger Flow*

4.4.2 Collaboration View

The *Collaboration View* defines the e-business collaborations between the modelled enterprise and its trading partners. It describes the contract established between the participants. In the contract, each enterprise commits itself to trigger an economic event in exchange of another economic event. These events consist in transferring a resource to the other partner. As you can see, the Collaboration View reuses the principles behind REA (which is described in point 3.4.2).

The *Collaboration View* consists of the following modelling elements:
- *Business Collaboration*
- *Economic Commitment*
- *Economic Contract*
- *Economic Event*
- *Economic Resource*
- *Economic Resource Type*
- *Partner Type*

4.4.3 Interaction View

The *Interaction View* describes all the interactions needed to execute a *Business Collaboration* and their choreography. For each interaction, it defines the role of the participating partners.

The *Interaction View* consists of the following modelling elements:
- *Business Collaboration*
- *Business Interaction*
- *Partner Type*

4.4.4 Information Exchange View

The *Information Exchange View* describes a *Business Interaction* step by step and shows the exchanged *Business Information* between the trading partners.

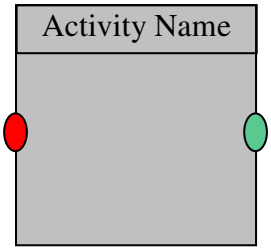
The *Information Exchange View* consists of the following modelling elements:
- *Business Information*
- *Business Interaction*
- *Partner type*

4.5 Graphical concrete syntax

This section describes the graphical elements used to model with EBCML. As explained in the previous section, the modelling technique uses four different views. Every view has its own modelling elements and thus its own graphical syntax. A Microsoft Visio stencil has been created for each view.

4.5.1 Enterprise View

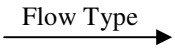
Modelling Elements



An *Activity*

An *Input Port*

An *Output Port*



A *Flow*. It can be *IO flow*, Re flow (*Resource Flow*), Cl flow (*Control Flow*), Tr Flow (*Trigger Flow*), Ct Flow (*Constraint Flow*)

An "AND" *Connection Operator*

An "OR" *Connection Operator*

A "XOR" *Connection Operator*

A *Human Resource*. "Worker" is only an example of human resource.

A *Material Resource*. "Product" is only an example of material resource.

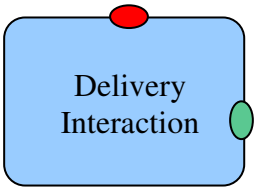
An *Economic Resource* (it can be a product, a service or money)

An *Information Object*

A "person" *Resource Role Type* (an employee of the modelled enterprise). {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}

An "orgunit" *Resource Role Type* (a group of people from the modelled enterprise or another company with which we collaborate but do not make e-business). {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}

A "tool" resource role type {This is an attribute of the class *Role Type* in the meta-model. For readability reasons, we have chosen to represent it with a unique symbol.}



An *Interaction* between the modelled company and a business partner. Each collaboration contains at least one interaction. All the interactions composing a collaboration are identified in the *Interaction View* and described in the *Information Exchange View*.

Example

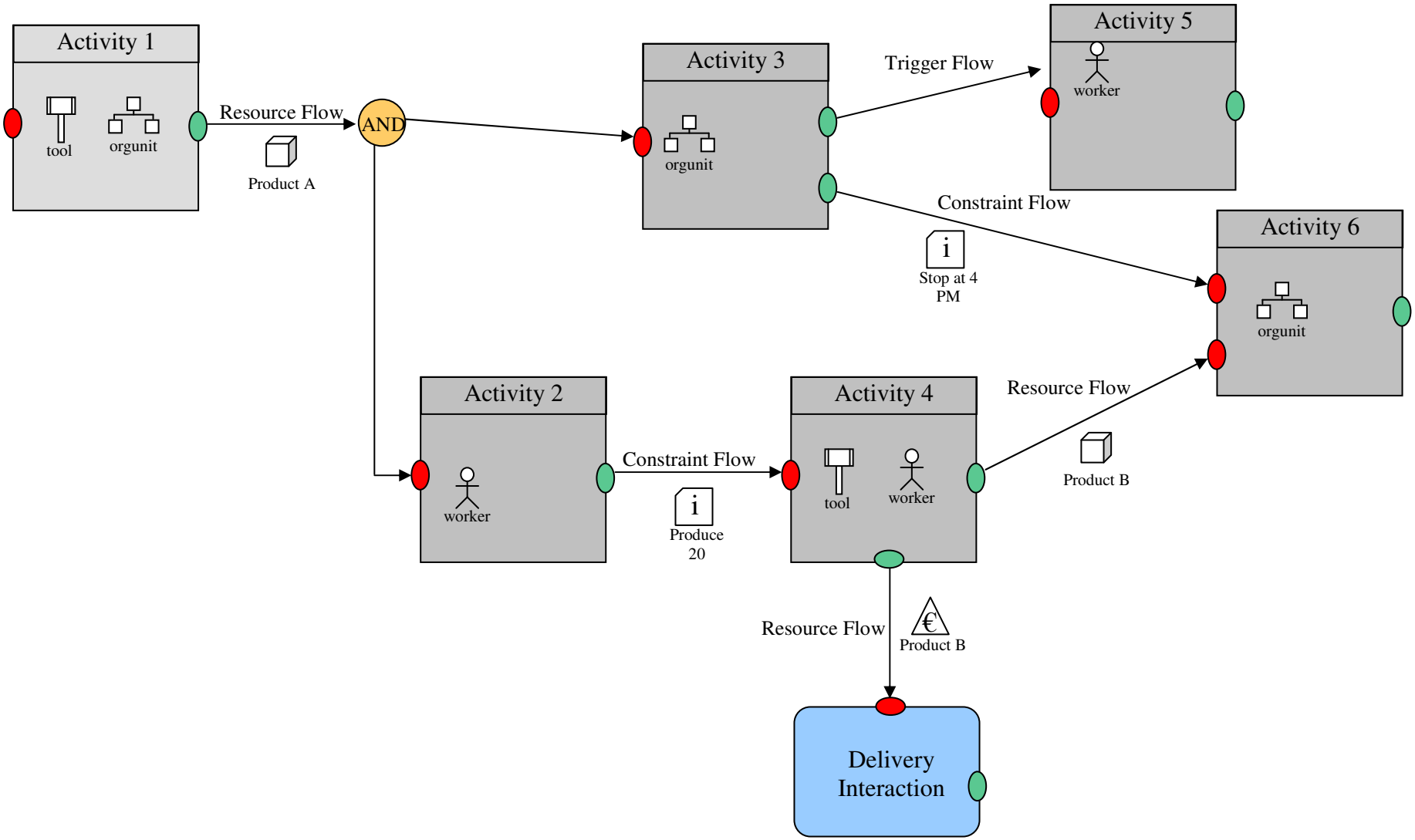


Figure 4-3: Example of an Enterprise View model

Screenshot while using the dedicated Microsoft Visio stencil

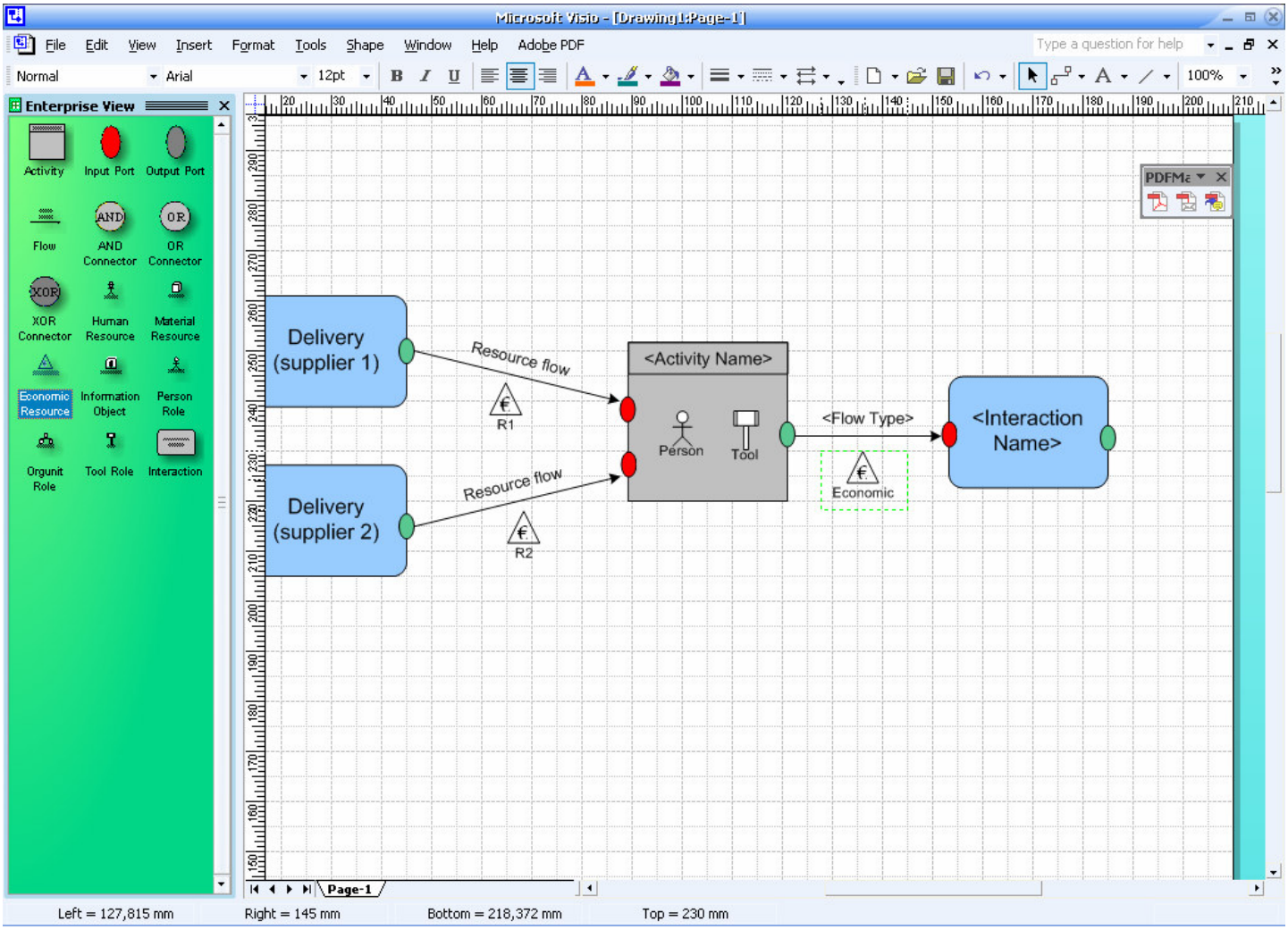
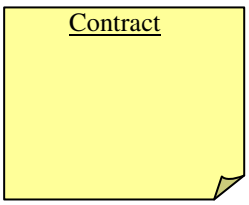


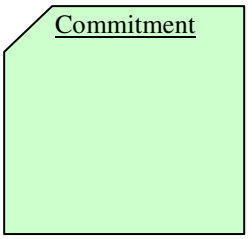
Figure 4 - 1: Modelling in EBCML with Microsoft Visio: Enterprise View

4.5.2 Collaboration View

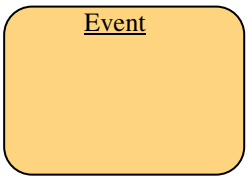
Modelling Elements



An *Economic Contract*



An *Economic Commitment*



An *Economic Event*

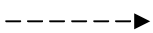
A *Business Partner Type* involved in the collaboration

Economic Resource Type: money

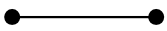
Economic Resource Type: material

Economic Resource Type: service

Participates



Specifies



Fulfills

Example

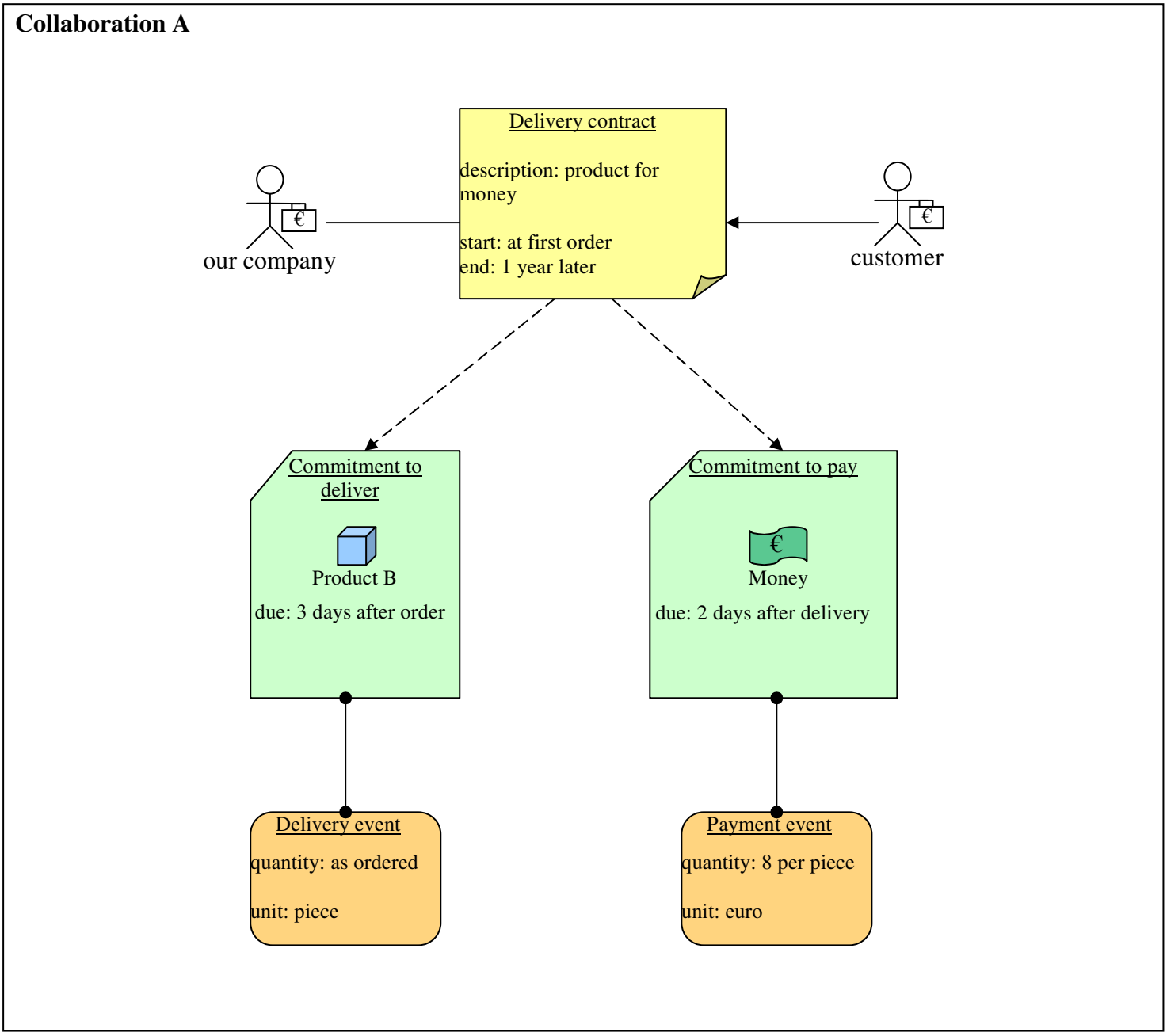


Figure 4 - 2: Example of a Collaboration View model

Screenshot while using the dedicated Microsoft Visio stencil

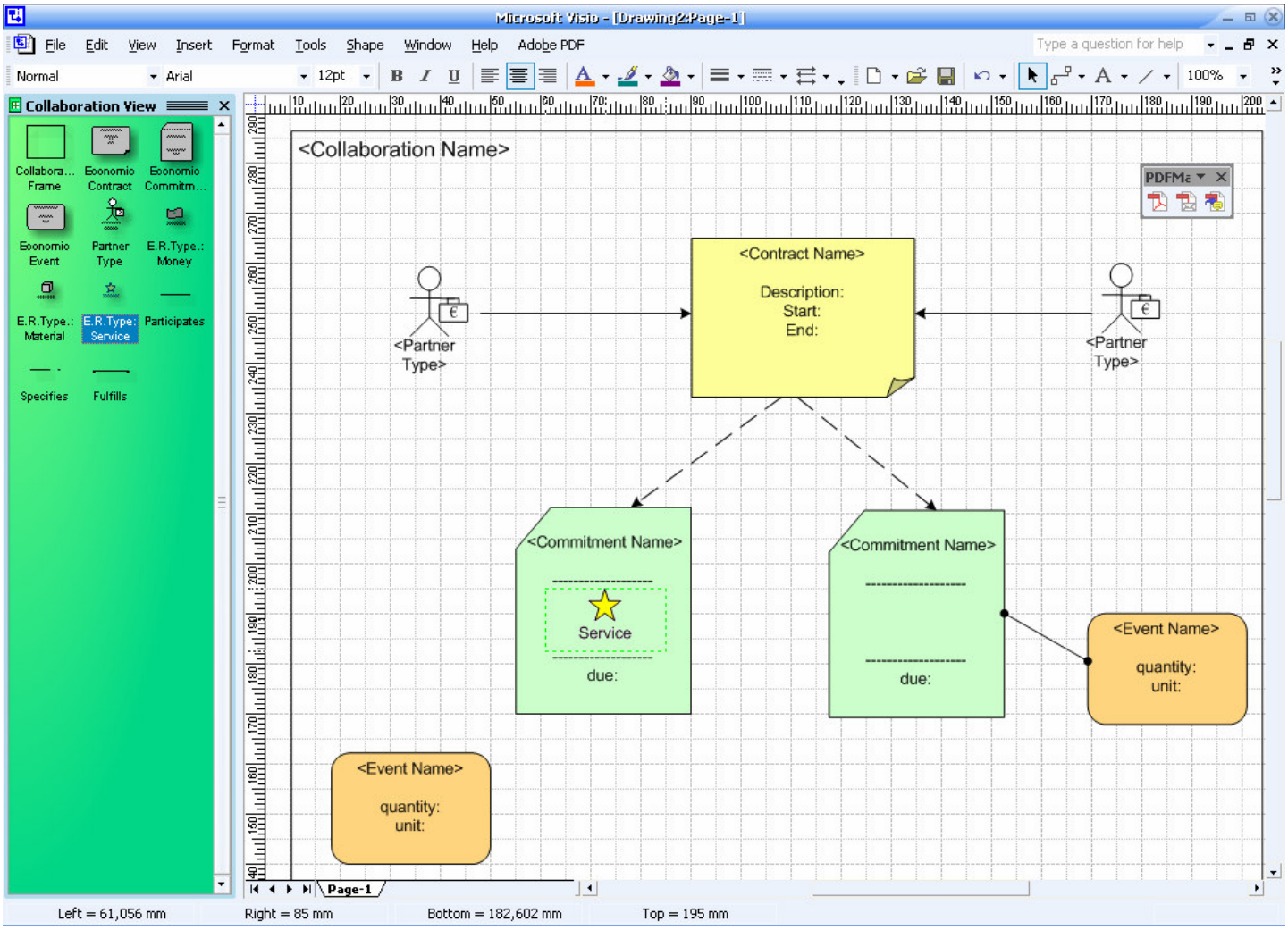
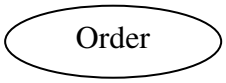


Figure 4 - 3: Modelling in EBCML with Microsoft Visio: Collaboration View

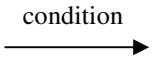
4.5.3 Interaction View

Modelling Elements



A *Business Interaction*

A *Business Partner Type* involved in the interaction



"Causes" relationship: orders the execution of interactions (the condition is optional)

Example

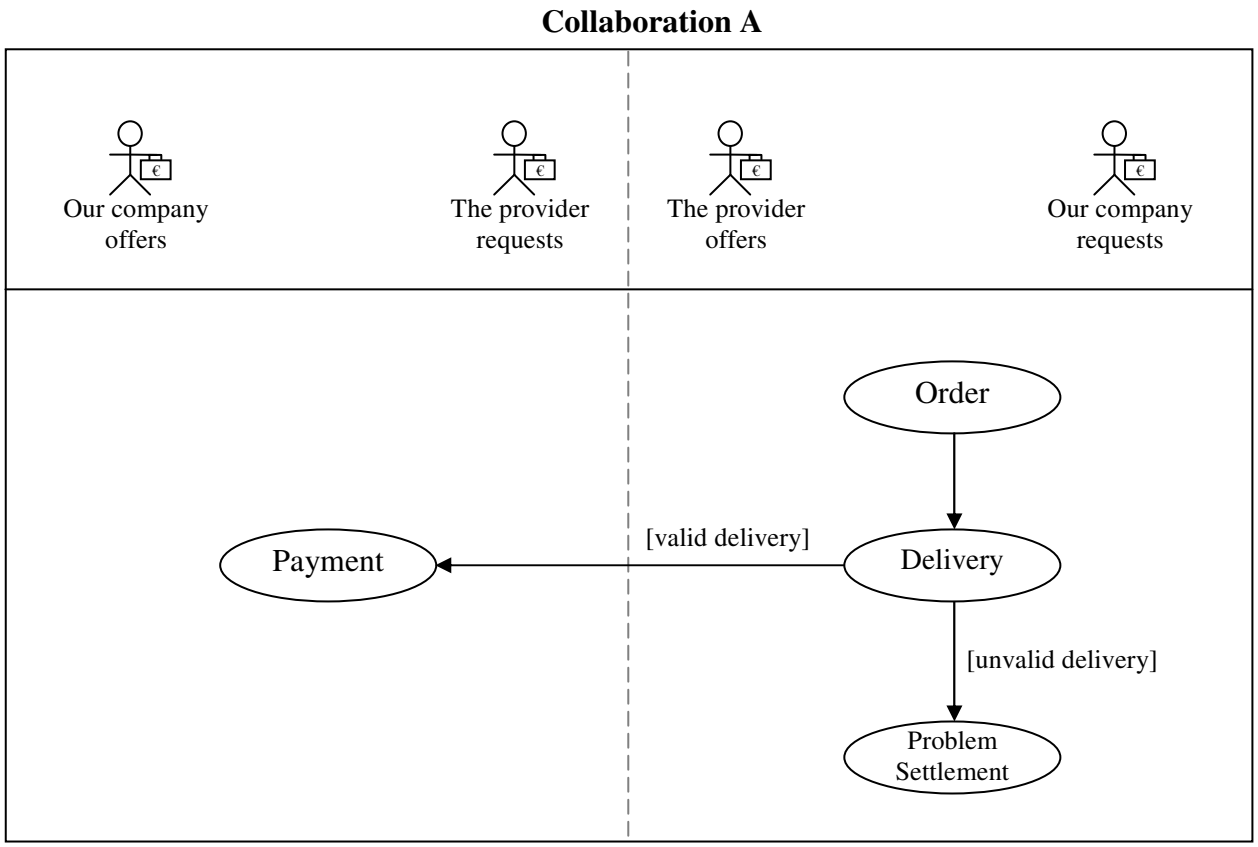


Figure 4 - 4: Example of an Interaction View model

Screenshot while using the dedicated Microsoft Visio stencil

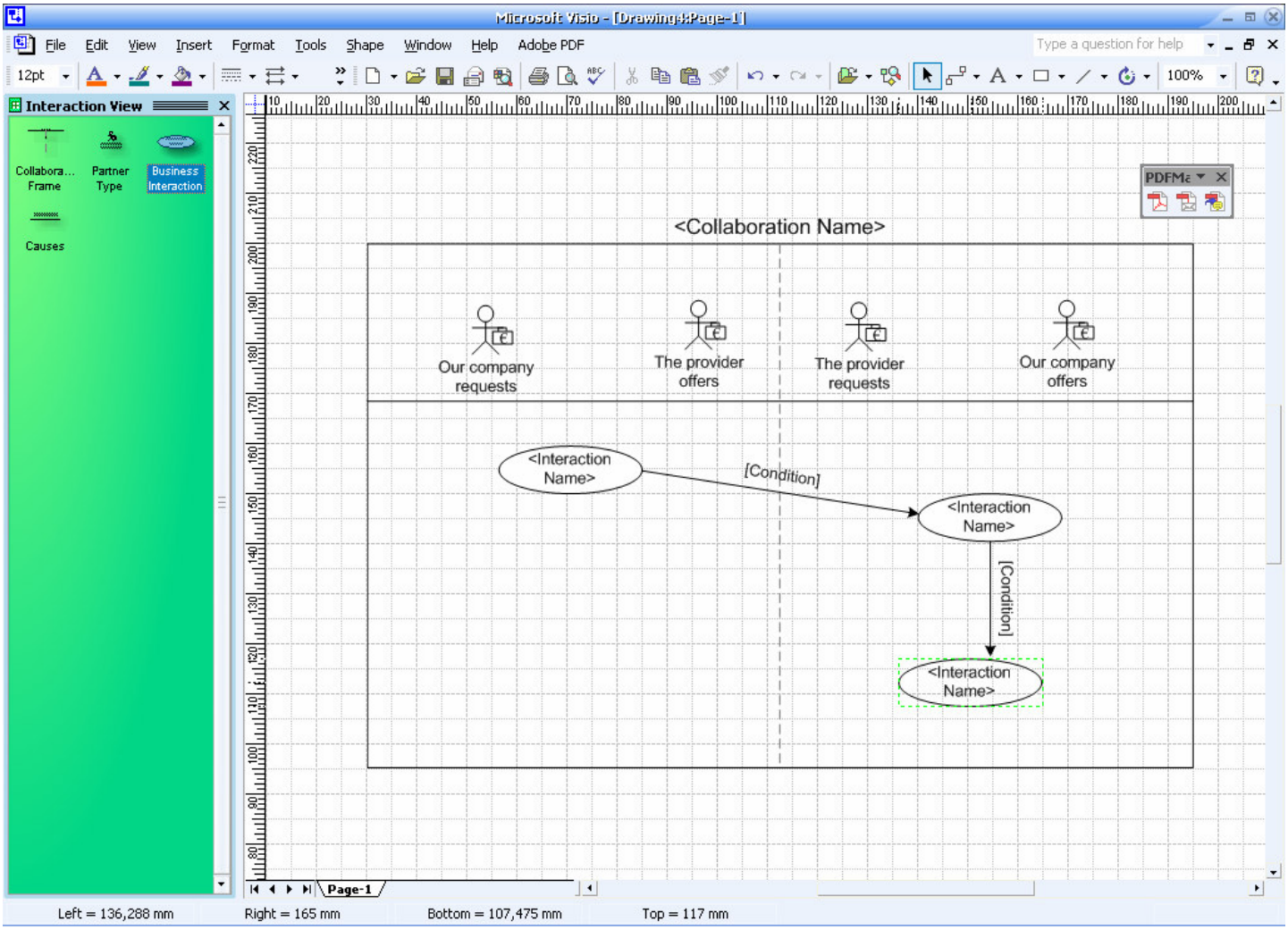


Figure 4 - 5: Modelling in EBCML with Microsoft Visio: Interaction View

4.5.4 Information Exchange View

Modelling Elements

 Our logistics manager	A person (representing a partner type)
 The provider's system	An information system (representing a partner type)
<Ordering Information>	A *Business Information* (optionally electronic (XML, etc.))
and	Operator used to compose actions
or	Operator used to indicate a choice between actions

Example

Interaction "Order"

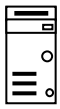
 Our logistics manager	 The provider's system
1. Connect 3. Send <Customer ID> 5. Send <Ordering Information> 7. Disconnect	2. Ask for <Customer ID> 4. Ask for <Ordering Information> 6. Accept order and Send <Ordering ID>
7'. Analyse <Ordering Failure Report> 8. Update <Ordering Information> or Disconnect	6' Refuse order and Send <Ordering Failure Report>

Figure 4 - 6: Example of an Information Exchange View model

Screenshot while using the dedicated Microsoft Visio stencil

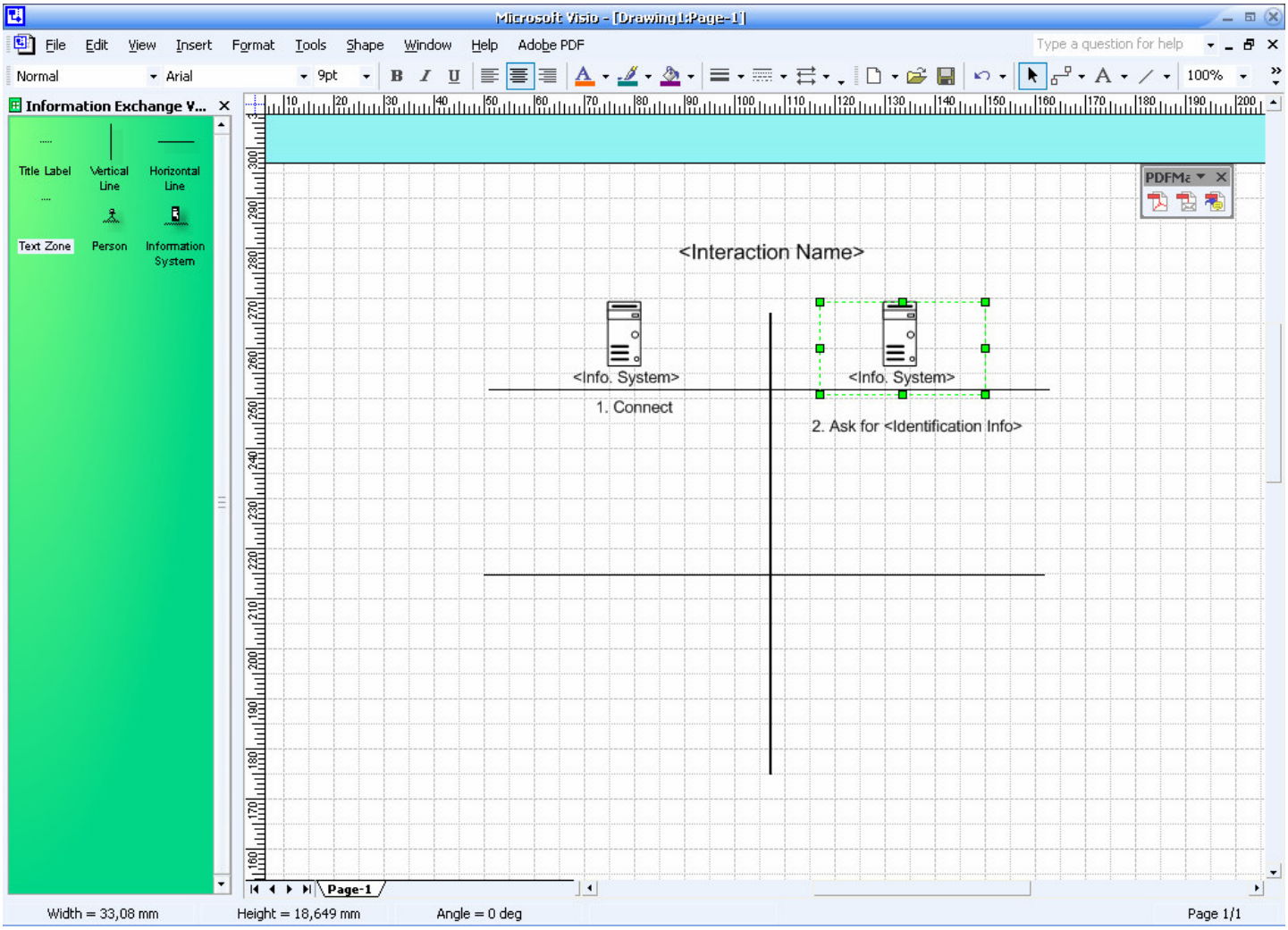


Figure 4 - 7: Modelling in EBCML with Microsoft Visio: Information Exchange View

4.6 Case study

Description

This simple case study will depict how we can model a business domain with EBCML. Let us imagine a simplified business net involving five partners.

The first partner, CoolDrinks[1], is a soda, lemonade and orangeade provider. The business activity of this enterprise can be summarized as follows:

- it buys fruit and sugar from providers;
- it produces different kinds of drinks (soda, lemonade, orangeade, etc.);
- it puts the drinks produced in bottles;
- it packages the bottles;
- it delivers the packages to supermarkets.

The second partner, BigFruit, is a fruit provider. It harvests fruit in different parts of the world, treats them for conservation and delivers them as quickly as possible to its customers.

The third partner, SuperSugar, is a sugar producer. It manages a refinery and sells the produced sugar to different sorts of customers.

The fourth partner, StarMarket, is a supermarkets chain. It buys a huge amount of products from different providers and sells them directly to private individuals.

The fifth partner, UniversalBank, is a bank company. It provides all the classic financial services: account information management, money transfers, solvability warranty, etc.

There are several business collaborations in that business net:

- the fruit needed by CoolDrinks is provided by BigFruit;
- the sugar needed by CoolDrinks is provided by SuperSugar;
- the drinks sold by StarMarket are provided by CoolDrinks;
- the fruit sold by StarMarket is provided by BigFruit;
- the sugar sold by StarMarket is provided by SuperSugar;
- the bank services that CoolDrinks, BigFruit, SuperSugar, and StarMarket use are provided by UniversalBank.

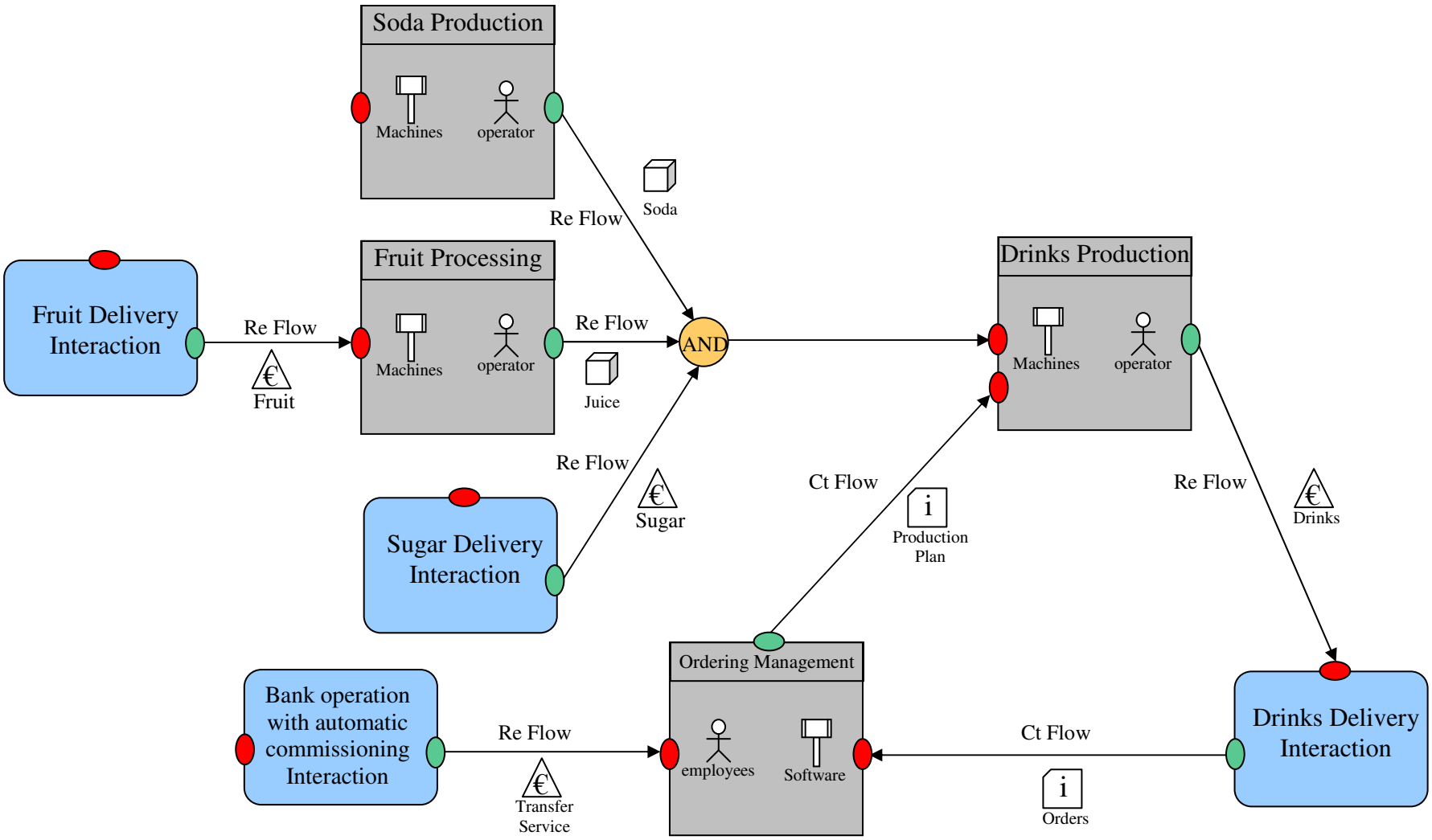
In the following pages, we will use EBCML to model the internal processes of CoolDrinks and the business collaborations in which this enterprise is involved. The study of this single enterprise is enough to have an idea of how EBCML can be used.

The EBCML models of the four other enterprises (BigFruit, SuperSugar, StarMarket, and UniversalBank) can be found in Annex 4.

[1] All the enterprises mentioned here are fictive and do not refer to any real company even if, by any chance, the names are identical.

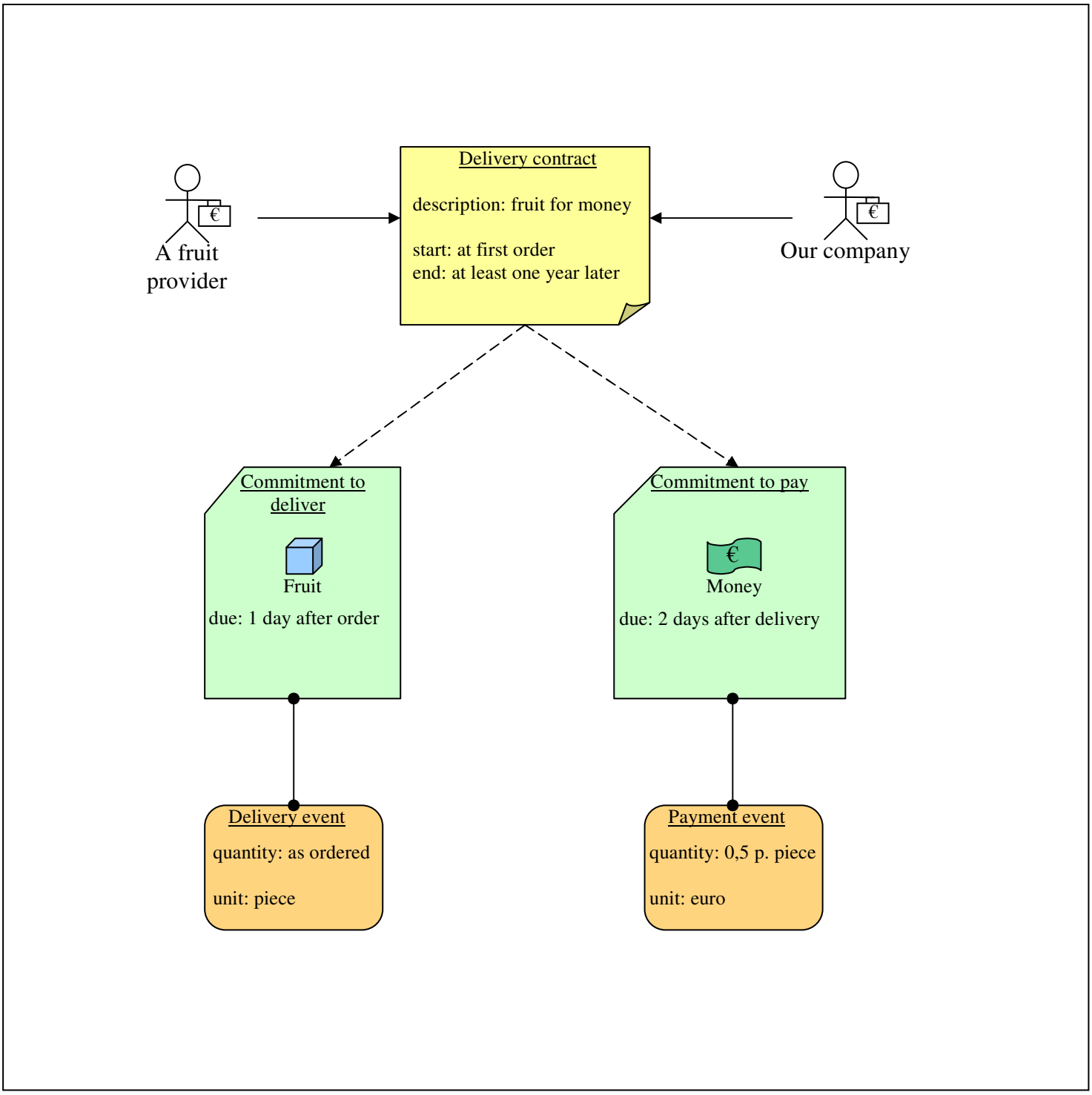
CoolDrinks

Enterprise View

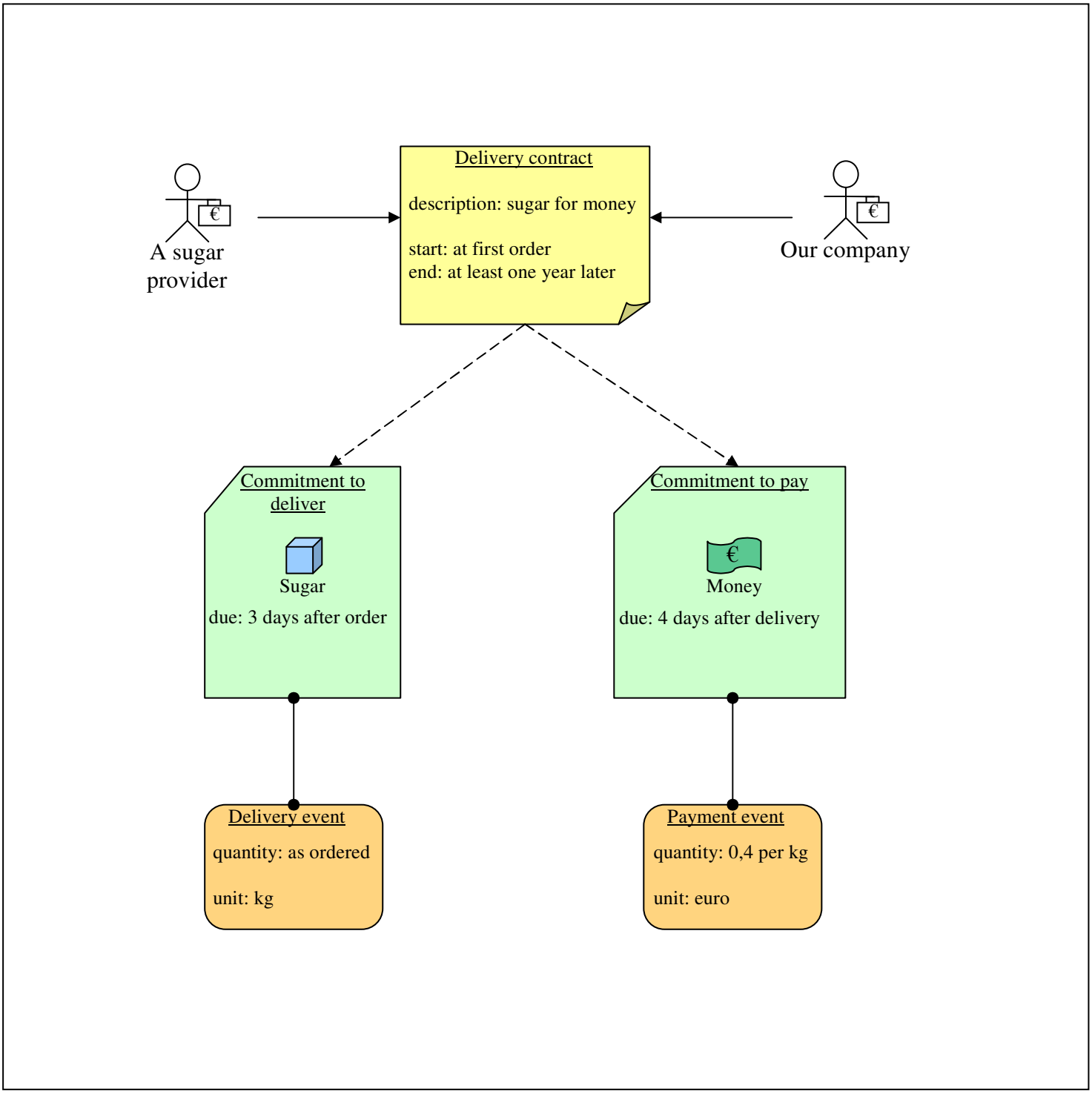


Collaboration View

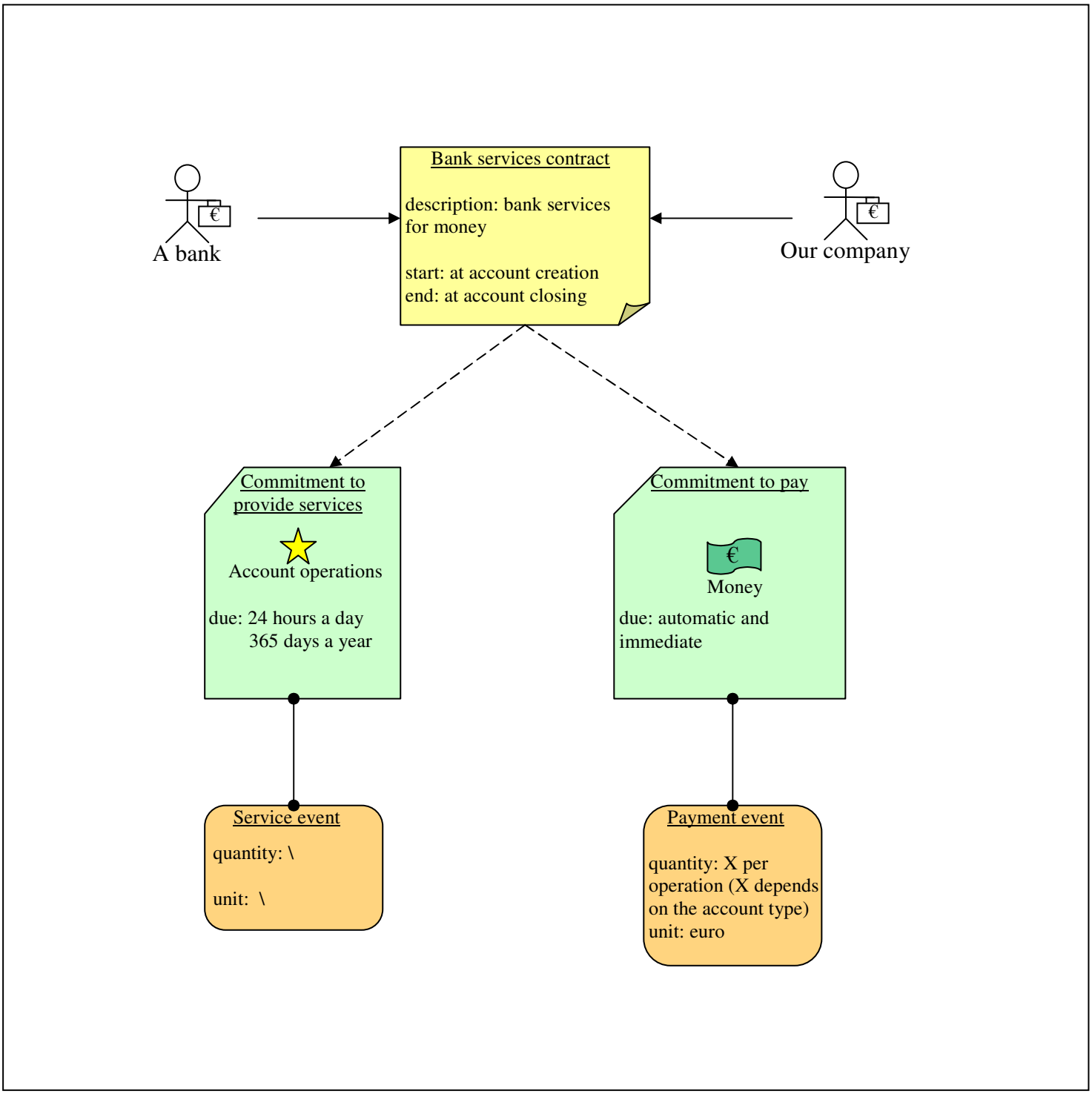
Fruit Delivery Collaboration



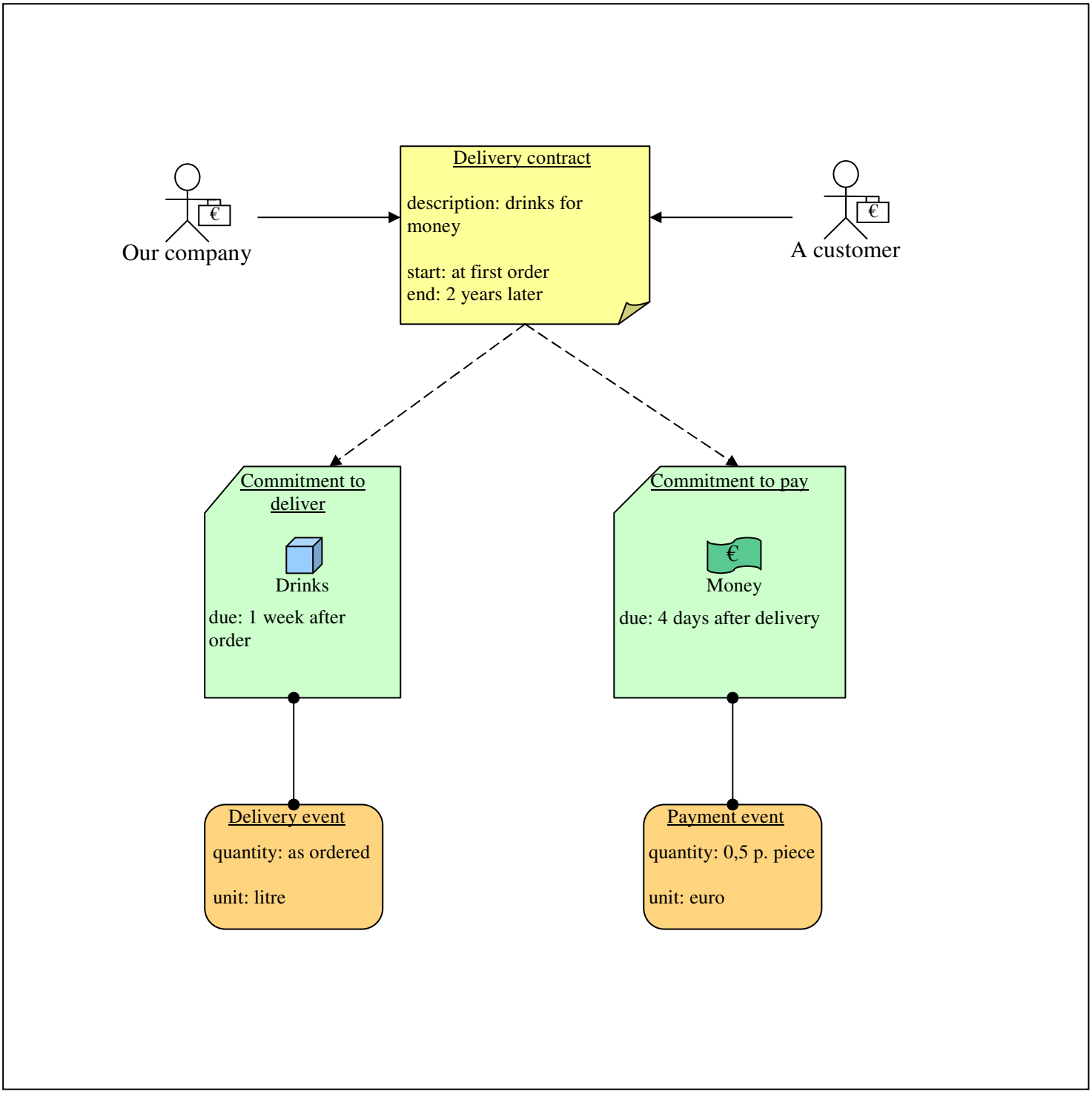
Suger Delivery Collaboration



Bank Account Collaboration

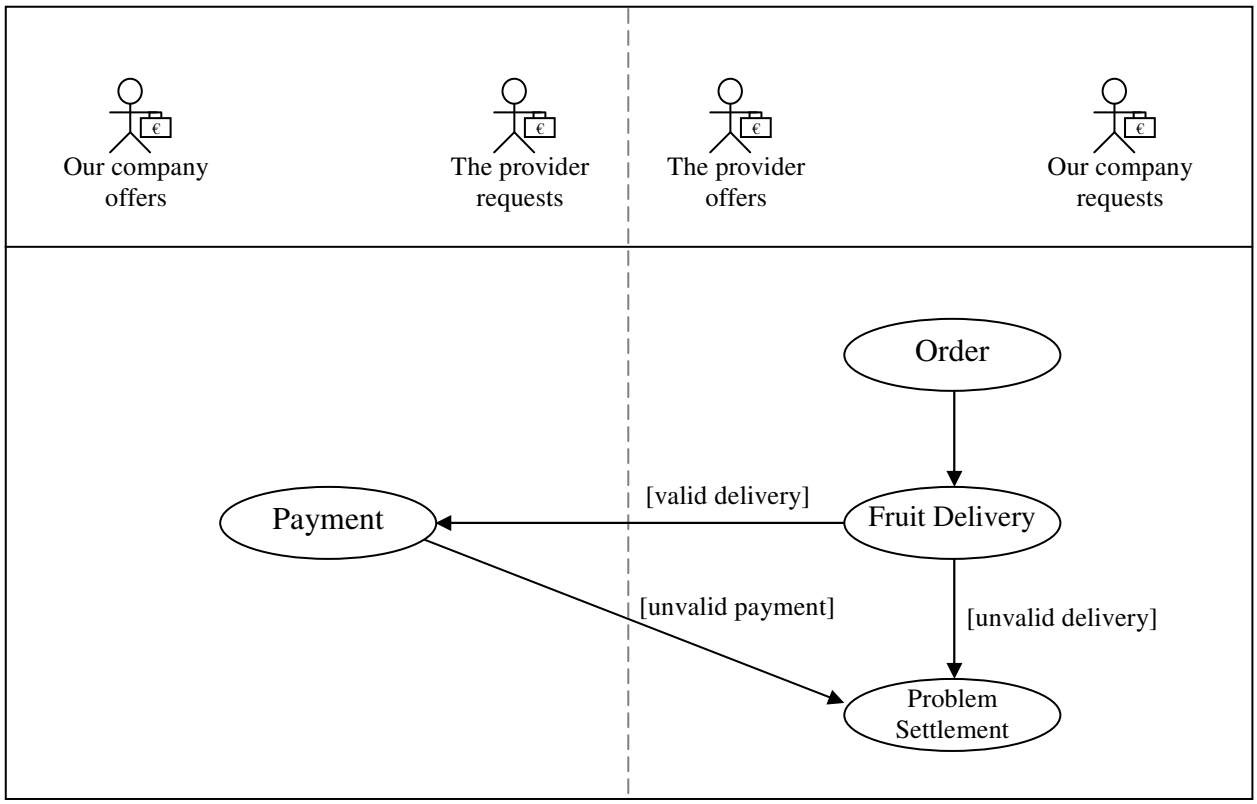


Drinks Delivery Collaboration

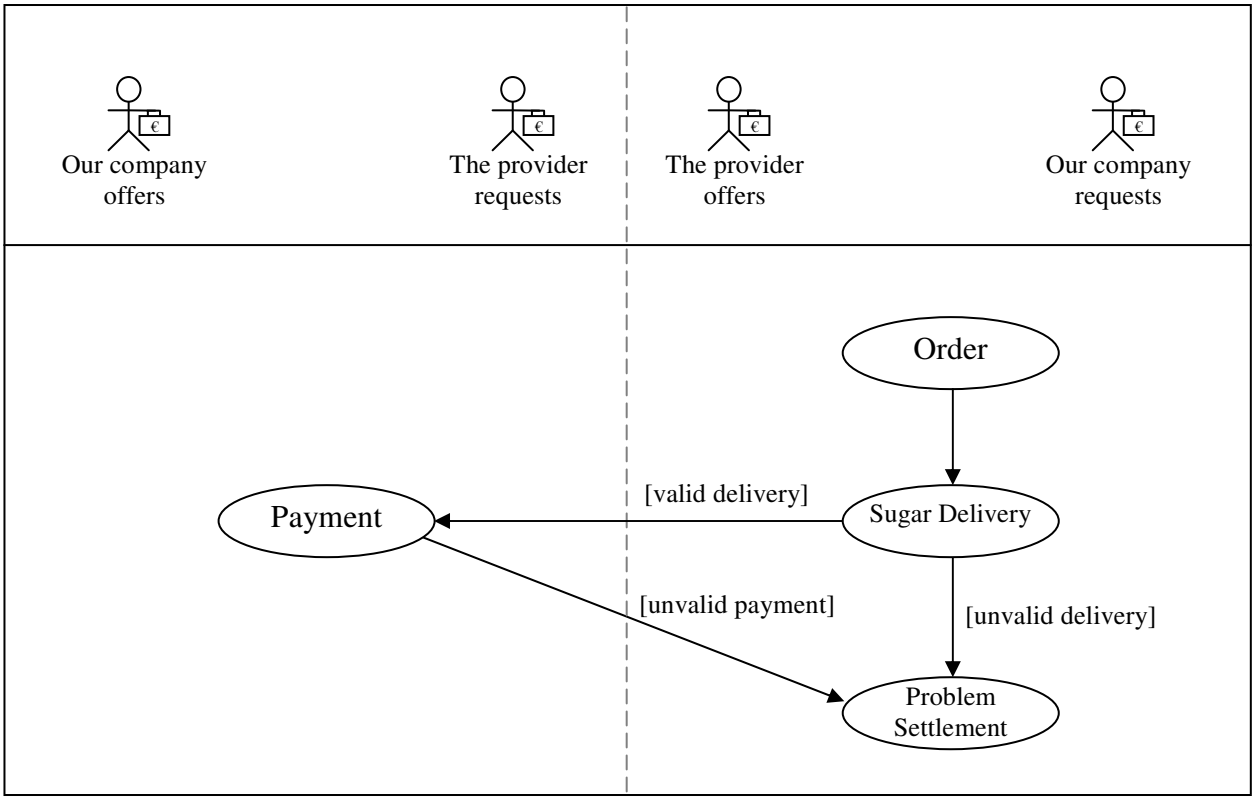


Interaction View

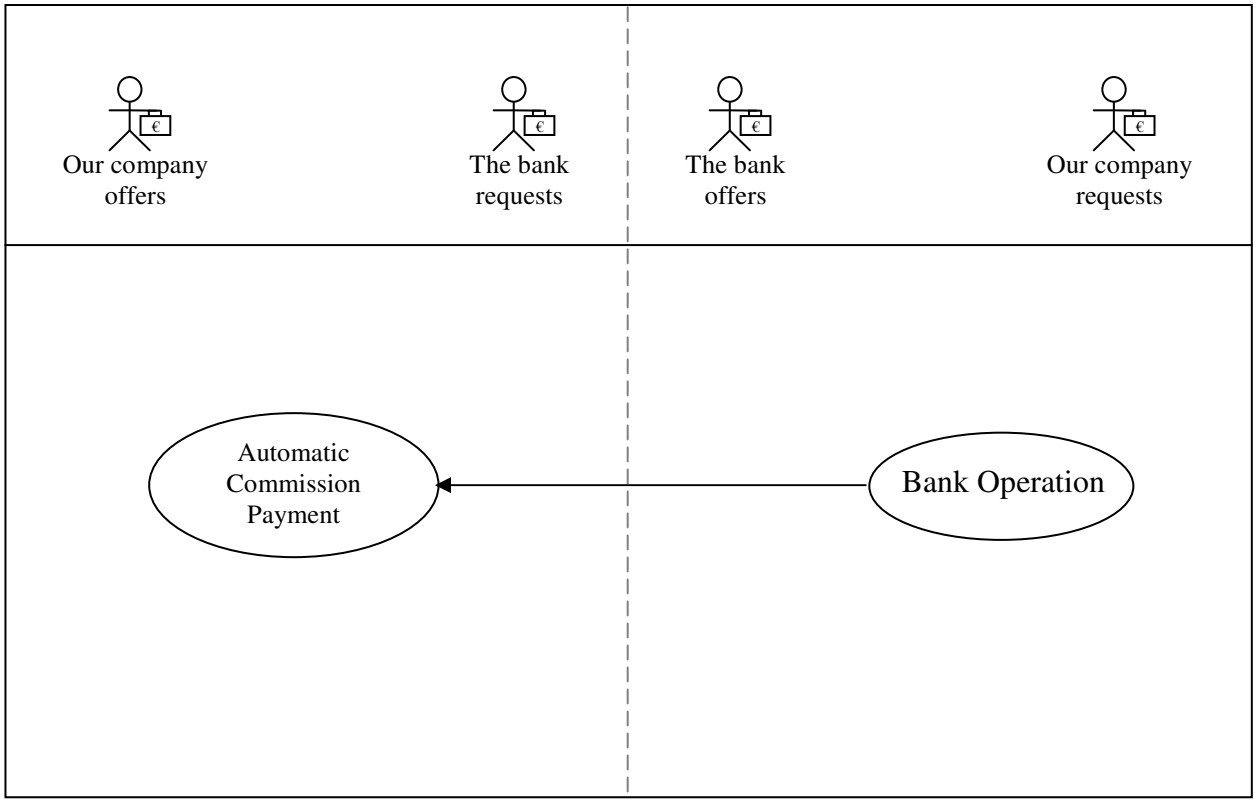
Fruit Delivery Collaboration



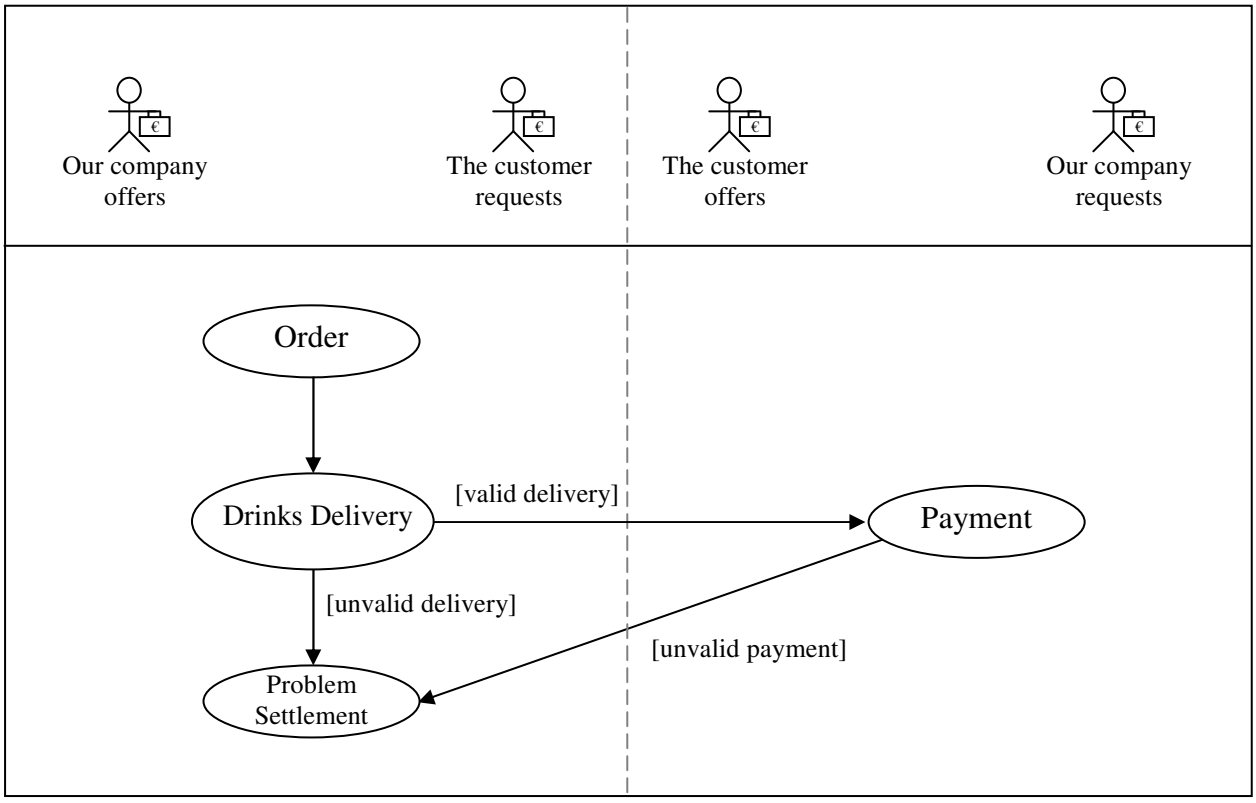
Sugar Delivery Collaboration



Bank Account Collaboration



Drinks Delivery Collaboration



Information Exchange View

Here are described several information exchanges. Those not described are nearly identical to those provided here.

Interaction "Order"

 Our logistics manager	 The provider's system
1. Connect 3. Send <Customer ID> 5. Send <Ordering Information> 7. Disconnect	2. Ask for <Customer ID> 4. Ask for <Ordering Information> 6. Accept order and Send <Ordering ID>
7'. Analyse <Ordering Failure Report> 8. Update <Ordering Information> or Disconnect	6' Refuse order and Send <Ordering Failure Report>

Interaction "Fruit Delivery"

 Our logistics manager	 The fruit deliverer
2. Check delivery with contract 3. Accept delivery 6. Sign <Receipt> 7. Give <Signed Receipt>	1. Deliver fruit 4. Give <Invoice> 5. Give <Receipt>
3'. Refuse delivery	4'. Leave with freight

Interaction "Payment"

 Our financial manager	 The provider's financial manager
1. Pay for the delivery	2. Check payment with contract 3. Accept payment 4. Send <Acknowledgement>
	3'. Refuse payment

Interaction "Drinks Delivery"

 Our drinks deliverer	 The cusomer's logistics manager
1. Deliver drinks 4. Give <Invoice> 5. Give <Receipt>	2. Check delivery with contract 3. Accept delivery 6. Sign <Receipt> 7. Give <Signed Receipt>
4'. Leave with freight	3'. Refuse delivery

Interaction "Problem Settlement"

 Our mediator	 The provider's mediator
1. Send <Complaint over delivery>	
	2. Analyse <Complaint over delivery> 3. Propose a solution
4. Accept solution proposed or file a lawsuit against the provider	
	1'. Send <Complaint over payment>
2'. Analyse <Complaint over payment> 3'. Propose a solution	
	4'. Accept solution proposed or file a lawsuit against the customer

Interaction "Bank Operation"

 Our financial manager	 The bank's system
1. Connect	
	2. Ask for <Customer ID>
3. Send <Customer ID>	
	4. Ask for <Bank Operation Information>
5. Send <Bank Operation Information>	
	6. Execute operation
7. Disconnect	

Interaction "Automatic Commission Payment"

 Our financial manager	 The bank's system
	1. Deduct commission from customer's account 2. Update <Customer's Account Information>
3. Check regularly <Customer's Account Information>	

5. Conclusion and possible extensions to this work

The goal of this thesis was twofold. The first objective was to identify the most important concepts in both enterprise modelling languages and e-business modelling languages. The second goal was to propose a modelling language able to model both aspects.

In this work, we have focused on the study of two languages: UEML (the Unified Enterprise Modelling Language), which is a good representative of enterprise modelling languages, and UMM (the UN/CEFACT Modelling Methodology), which is a good representative of e-business modelling languages. Once the most interesting[1] UMM concepts have been identified, we have added them to UEML in order to extend this language by introducing e-business modelling constructs. In doing so, we have obtained a new version of UEML that we have called EBCML (the Enterprise and Business Collaboration Modelling Language). EBCML effectively allows to model both intra-enterprise processes *and* business collaborations. So, we are now able to say that it is possible to describe both aspects with a single modelling language. However, the creation of such a language was not easy. The difficulty came mainly from the difference between the objectives of UEML and UMM. It was necessary to establish a bridge between the two worlds. For this, we had to introduce a keystone concept (the Business Interaction) to make the link. After having described the metamodel of the language, we have provided a graphical concrete syntax for it. Obviously, it would have been impossible to model anything without that. We have also created a prototype of editor for this graphical syntax in Microsoft Visio. Finally, we have shown the use of EBCML in the context of a simple case study.

We would like to stress that this thesis does not claim to provide a complete, finished EBCML. The objective was rather to prove the feasibility of such a language. Several extensions to this work are possible.

First of all, there is currently no formal semantics for EBCML. Indeed, the semantics is described in natural language and illustrated by a case study. Actually, many enterprise modelling languages do not have a formal semantics. There are two main reasons for this. Firstly, it is a complicated matter and secondly, there is a problem of consensus. However, formal semantics make (semi-) automatic processing of models possible. This semantics would be useful for mappings between EBCML and other languages, including other modelling languages but also e-business applications languages.

Secondly, one of the goals of EBCML is to facilitate the development of e-business software. This thesis has focused on the modelling part but it would be interesting to use EBCML models in order to create an entire e-business system. Existing e-business frameworks, such as ebXML, could be used to that end. A thesis could validate the proposed modelling approach by treating an entire case study, from the requirements and their expression in EBCML to the implementation of the system.

Thirdly, a complete modelling tool could be developed for EBCML. In this work, we have customized Microsoft Visio in order to model with EBCML. However, Microsoft Visio is not the most appropriate tool. The use of the proposed stencils is strictly based on a drag and drop scheme. The user is supposed to know everything about EBCML and the relevance of what is done is not checked. The user can do what he/she wants and is not compelled to respect the syntax. A more complete and user-friendly tool would be appreciable.

[1] In terms of collaboration modelling.

6. Bibliography

The web links provided in this bibliography were used between 06/06/2004 and 04/30/2005. The author cannot guarantee their ulterior availability.

[Vernadat 1996] François B. Vernadat, “Enterprise Modeling and Integration Principles and applications”, Chapman & Hall, pp. 6-20, 1996

[Chappell&al. 2001] David A Chappell, Vivek Chopra, Jean-Jacques Dubray, Colleen Evans, Betty Harvey, Tim McGrath, Duane Nickull, Marcel Noordzij, Bruce Peat, Pim van der Eijk and Jan Vegt, “Professional ebXML foundations”, Wrox Press, pp. 44-46, 2001

[Harel&Rumpe 2000] David Harel and Bernhard Rumpe, "Modeling Languages: Syntax, Semantics and All That Stuff", 2000.

[Berio&Petit 2003] Giuseppe Berio and Michaël Petit, "Enterprise modelling and the UML: (sometimes) a conflict without a case", 2003.

[UML site] The UML official site
http://www.uml.org/

[Metamodel site] Community site for meta-modeling and semantic modeling
http://www.metamodel.com/

[UMM-N090R10] The UMM methodology description
http://www.unece.org/cefact/tmg/UMM-N090R10-ALL.zip

[UMM UG] The UMM User Guide
http://www.unece.org/cefact/tmg/umm_ug_v030922.pdf

[Hofreiter&al. 2004] Birgit Hofreiter, Christian Huemer and Werner Winiwarter, “OCL-constraints for UMM Business Collaborations”, 2004
http://www.pri.univie.ac.at/~winiwarter/ecweb04.pdf

[McCarthy 1982] William E. McCarthy, “The REA Accounting Model: A Generalized Framework for Accounting Systems in A Shared Data Environment.”, The Accounting Review (July), pp. 554-578, 1982

[REA site] A presentation of REA by its author, William E. McCarthy
http://www.msu.edu/user/mccarth4/cookie--elmo--basic%20REA.ppt

[Gruber 1993] Thomas Gruber, “A Translation Approach to Portable Ontologies,” Knowledge Acquisition, pp. 199-220, 1993

[UEML Org] The official site dedicated to the UEML project
http://www.ueml.org

[Berio UEML D3.1] Giuseppe Berio, "The UEML deliverable 3.1: Initial Core Constructs"
http://www.rtd.computas.com/websolution/FileRepository/Attachment_Web_DownLoad.asp?FileName=239_30_124__Deliverable_31.pdf

[UEML D3.1 An.] UEML project contributors, "Annexes 3.1"
http://www.rtd.computas.com/websolution/FileRepository/Attachment_Web_DownLoad.asp?FileName=239_30_124__Annexes31.zip

[CEN] European Committee for Standardization. Documents "EEG1 Invoicing Business Process v1.0.doc" and "EEG1 Invoice v1.0.doc"
http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/activity/eeg1brs.asp

7. Annexes

Annex 1. UMM Glossary

Hereafter follow the definitions of the most important UMM concepts. For certain concepts, there are several definitions, each coming from a different source. The reader will notice that these definitions are sometimes not consistent or even contradictory. This really complicates the understanding of UMM and we will tackle this problem in the review of UMM at the end of this chapter.

Agreement
"An agreement is an arrangement between two partner types that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific economic commitments." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Action
"The state of a business transaction is defined by reciprocal Business Actions executed by an authorized role. This is an abstract class that is not a stereotype." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Area
(1): "An area of knowledge or activity characterized by a family of related systems; an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area" ([UMM-N090R10], Annex1_UMM_Glossary).
(2): {The Business Areas subdivide the Business Domain Model. They represent the structure of the enterprise or of a general framework of a sector of activity. It is generally a market segment or an important operational division. A Business Area can be subdivided in sub-Business Areas or in Process Areas.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Collaboration
(1): "A business collaboration model specifies the input and output relationships between business collaboration use cases and Agents." ([UMM-N090R10], Chapter 8 MetamodelR10)
(2): {A business collaboration activity is a predefined set of activities of partners that is initiated by a partner to accomplish an explicitly shared business goal and terminated upon recognition of one of the agreed conclusions by all the involved partners. Business collaboration activities are specified by a business analyst as business processes, requirements and business object flow graphs that define the choreography of atomic business processes, referred to as Business Transactions.} (Our interpretation of the sometimes ambiguous [UMM UG])
(3): "A business collaboration is performed by two (= binary collaboration) or more (multi-party collaboration) business partners. A business collaboration might be complex and involve a lot of activities between business partners. However, the most basic business collaboration is a binary collaboration realized by a request from one side and an optional response from the other side. This simple collaboration is called *business transaction*." [Hofreiter&al. 2004]

Business Collaboration Use Case
"A business collaboration use case is an abstraction for a business collaboration protocol use case and a business transaction use case. The abstraction permits the reuse of the business collaboration realization relationship. A completed use case assumes that some one "thing" of "measurable value" be created either as a service performed or a product created." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Collaboration Protocol
(1): "A business collaboration protocol choreographs one or more business transaction activities." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {A Business Collaboration Protocol is composed of Business Transactions (each having states, preconditions, and postconditions). They are represented by an activity graph. The Business Collaboration Protocol defines the transitions between Transaction Activities. It is based on the states of the entities. Therefore, it defines the choreography of the whole collaboration. Each activity from the Business Collaboration Protocol is a Transaction Activity and is further detailed by a Transaction. For each Transaction Activity, there is exactly one Transaction (and vice-versa). These two notions are synonyms in the business world but are modelled differently.} (Our interpretation of the sometimes ambiguous [UMM UG])
(3): "Since UMM is based on UML, it uses the concept of use cases to capture requirements. In case of a complex business collaboration the requirements are described in a so-called *business collaboration protocol use case*. These requirements lead to a choreography of activities in order to create the customer value. The activity graph representing this choreography is called *business collaboration protocol.* Each activity shown in a business collaboration protocol refers to exactly one business transaction. Therefore, each activity of the business collaboration protocol is called a *business transaction activity*." [Hofreiter&al. 2004]

Business Collaboration Protocol Use Case
"A business collaboration protocol use case is used to gather requirements for e-business collaboration protocol specifications." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Document
A Business Document is a set of information that has business significance and is exchanged between Business Partners. It is built by customizing and combining Business Information.

Business Domain View (BDV)
(1): {The BDV is used to create the Business Domain Model. This model is divided into Business Areas, Process Areas and Business Processes. The role of this view is to identify the Business Processes that need an e-business collaboration. It allows to know if it is possible to reuse existing descriptions in the UMM libraries.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Entity
(1): "Something that is accessed, inspected, manipulated, produced, and so on in the business." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): "Business Entity is an abstraction for any artifact that is important in the execution of a business collaboration." ([UMM-N090R10], Chapter 8 MetamodelR10)

Examples of Business Entities include an invoice, an order for a product, etc.

Business Information
{In a collaborative environment, partners must exchange Business Information to know the current states of the Business entities. Therefore, Business Information exchanges provoke changes in the states of Business Entities. The Business Information must indicate all the entities that change state following the exchange. Moreover, the Business Information contains a header with more general information (i.e. independent of the Business Entities).} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Object
{Business Information is manifested by Business Objects. A Business Object is a reusable class representing a particular business concept. In combining these objects, we are able to create business information structures. There exists a library containing a large amount of reusable Business Objects.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Process
(1): "A Business Process is a use case that is used to gather requirements about business processes." ([UMM-N090R10], Chapter 8 MetamodelR10)
(2): "The means by which one or more activities are accomplished in operating business practices." ([UMM-N090R10], Annex1_UMM_Glossary)
(3): {A *business process* is defined as an organized group of related activities[1] that together create customer value. If all the activities are performed by one organization this leads to an intraorganizational business process. In B2B[2], the activities are executed by different organizations which collaborate to create value. UMM focuses on interorganizational business processes and calls them *business collaborations*. A Business Process can divided into sub-Business Processes. An Activity can be viewed as } (Our interpretation of the sometimes ambiguous [UMM UG])

Business Requirements View (BRV)
(1): "The view of a business process model that captures the requirements of a business collaboration protocol." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {The BRV defines how the process to model is executed in real life. It allows to capture the business scenarios, inputs, outputs, constraints and boundaries for Business Processes and their interrelationships within business process collaborations. This view shows how the business domain expert sees and describes the process to be modelled. The BRV is expressed in the language and concepts of the business domain expert.} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Service View (BSV)
(1): "The view of a business process model that specifies the electronic formation of business contracts using an electronic medium." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {The BSV defines the services, the agents, and the needed messages to establish a Business Collaboration. This view uses technical concepts (i.e terms useful to software developers).} (Our interpretation of the sometimes ambiguous [UMM UG])

Business Transaction
(1): "A business transaction is a set of business information and business signal exchanges among two business partners that must occur in an agreed format, sequence and time period." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {A Business Transaction is an atomic collaboration. There exists six Transaction Patterns. Each Business Transaction must belong to one of these patterns. A business transaction involves sending business information from one partner to the other and an optional reply.} (Our interpretation of the sometimes ambiguous [UMM UG])
(3): "The most basic business collaboration is a binary collaboration realized by a request from one side and an optional response from the other side. This simple collaboration is a unit of work that allows roll back to a defined state before it was initiated. Therefore, this special type of collaboration is called *business transaction*. The requirements of a business transaction are described by a *business transaction use case*. The requirements lead to a choreography of the business transaction. The resulting activity graph is what is really called *business transaction* in UMM. One might argue that business transaction activity and business transaction present the same concept. Since different UML elements - an activity and an activity graph - are required in the UML notation, these concepts are distinguished in UMM. The activity graph of a business transaction is always composed of two *business activities*, an *initiating business activity* performed by the initiator and a *reacting business activity* performed by the other business partner. In a *one-way transaction,* business information is exchanged only from the *initiating business activity* to the *reacting business activity*. In case of a *two-way transaction,* the reacting business activity returns business information to the initiating business

[1] An actictivity is an atomic unit of work. It is the lowest-level of functionality within a business process.
[2] B2B stands for "Business To Business". It is about e-business collaborations between enterprises. To be simple, e-business can be divided into two major categories: B2B and B2C ("Business To Customer", which is about e-business collaborations with private individuals)

activity. In UMM we distinguish two one-way transactions (which are two different *transaction patterns*) - *notification* and *information distribution*- and four two-way transactions (which are four different *transaction patterns*) - *query/reponse*, *request/confirm*, *request/response* and *commercial transaction*. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi." [Hofreiter&al. 2004]

Business Transaction Activity
"A business transaction activity is a business collaboration protocol activity that executes a specified business transaction." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Transaction Use Case
"A business transaction use case is used to gather requirements for business transaction specifications." ([UMM-N090R10], Chapter 8 MetamodelR10)

Business Transaction View (BTV)
(1): "The view in a business process model that specifies the contract formation process for various types of business contracts." ([UMM-N090R10], Annex1_UMM_Glossary)
(2): {The view of a business process model that captures the semantics of business information entities and their flow of exchange between roles as they perform business activities. This view is an elaboration on the business requirements view by the business analyst and shows how the business analyst sees the process to be modelled. This view uses the language and concepts of the business analyst to convey requirements to the software designer and the business domain expert.} (Our interpretation of the sometimes ambiguous [UMM UG])

Information Entity
"An information entity realizes structured Business Information that is exchanged by partner roles performing activities in a business transaction. Information entities include or reference other information entities through associations. A secure information entity is an information entity with security controls. " ([UMM-N090R10], Chapter 8 MetamodelR10)

Partner Type
"A partner type is an actor in a business collaboration use case. Partner types are manufacturer, distributor, retailer, end user, carrier and financier. " ([UMM-N090R10], Chapter 8 MetamodelR10)

Process Area
(1): {A Process Area either subdivides the Business Domain Model in an orthogonal way to the Business Areas or subdivides a Business Area. A Process Area can be decomposed in sub-Process Areas or in Business Processes.} (Our interpretation of the sometimes ambiguous [UMM UG])

Requesting Business Activity
"A requesting business activity is a business activity that is performed by a partner role requesting business service from another business partner role." ([UMM-N090R10], Chapter 8 MetamodelR10)

Responding Business Activity
"A responding business activity is a business activity that is performed by a partner role responding to another business partner role's request for business service." ([UMM-N090R10], Chapter 8 MetamodelR10)

Annex 2. Definitions of the UMM meta-model classes and their attributes

(from [UMM-N090R10], Chapter 8 MetamodelR10)

BDV concepts

BusinessEntity
Business Entity is an abstraction for any artifact that is important in the execution of a business collaboration.

BusinessProcess
A business process is a use case that is used to gather requirements about business processes. Inputs to the business process must be specified in the preconditions and outputs from the business process must be specified in the post-conditions.
Tagged Values:

precondition. Preconditions are constraints that must be satisfied starting the use case.
beginsWhen. Describe the initial event from the actor that starts a use case.
definition. A set of simple sentences that state the
actions performed as part of the use case. This description includes references to "include" use cases and "extend" use cases.
endsWhen. Describe the condition or event that causes normal completion of the use case.
*exception*s. List all exception conditions that will cause the use case to terminate before its normal
completion.
postcondition. Post-conditions are constraints that must be satisfied ending the use case.
traceability. An explicit list of requirements, identified by requirements category, that are either partially or completely satisfied by this use case. Requirements categories are 1) Static and structural, 2) Dynamics, 3) Exception conditions, 4) Non-functional, 5) System Administration.

BusinessOperationsMap
A Business Operations Map is a framework for understanding business area sub-process nterrelationships. This framework is termed a Business Operations Map (BOM).
Tagged Values:

industrySegment. A specification of the scope of an industry-specific business process activity that encapsulates all of the business areas to be considered for the BOM
*businessOpportunity.*a statement of the business opportunity or the problem that is addressed by the BOM
references.

BusinessArea
A business area is a category of decomposable business process areas. A business area collates process areas.

BusinessCategory
A business category is an abstraction category for reusing tag-values. A business category collates sub-categories.
Tagged Values:

categorySchema. The name of the categorization schema used to reference use cases.
category. The category identifier used to reference a business area or process area set of use cases.
objective. A brief description of the purpose of the BOM, business area or process area

scope. A description of what the business area or process area applies to; what is affected or influenced by the business area or process area
boundary. A more detailed description of scope in terms of 1) stakeholders within/without, 2) information passed into or out from, 3) key business information objects used within, or 4) external interfaces to another - BOM, business area or process area

ProcessArea
A process area is a category of business processes and business transactions. A process area collates business processes and business transactions.

BusinessElement
Business element is an abstraction category for reusing tag- Values

Reference
If applicable, list documents that relate to the BOM or the business opportunities or problems which are to be addressed

StakeHolder
Represented by a role played in relation to the BOM, business area or process area

Constraints
Note any design constraints, external constraints or other dependencies

BRV concepts

BusinessCollaboration
A business collaboration model specifies the input and output relationships between business collaboration use cases and Agents. Agents provide input triggers to use cases and business collaboration use cases can provide input triggers and output triggers to and from other business collaboration use cases. A business collaboration model captures business information constraints imposed by a specific partner type collaboration. For example, sending a business document to a US Government agency requires a Standard Industry Classification (SIC) code to be included with the business information

BusinessCollaborationTask
A business collaboration task is a task that is performed by one business partner in collaboration with another business partner performing another business interface task. A business process is decomposed into business tasks and business interface tasks.
Tagged Values:
timeToPerform. A task is work that is performed with respect to time. There may be a specific time within which the task must be performed.

BusinessCollaborationProtocolUseCase
A business collaboration protocol use case is used to gather requirements for e-business collaboration protocol specifications.

BusinessCollaborationUseCase
A business collaboration use case is an abstraction for a business collaboration protocol use case and a business transaction use case. The abstraction permits the reuse of the business collaboration realization relationship. A completed use case assumes that some one "thing" of "measurable value" be created either as a service performed or a product created. Four appropriate classes of measure that can be applied to use case performance are: quantity measure, quality measure, time of performance measure and resource usage or consumption measure. Each use case should have an identified set of appropriate measures. At a minimum, at least one quantity measure should be employed.
Tagged Values:
recordMetrics.

BusinessProcessActivityModel
A business process activity model specifies the behavioural aspects of a business process. The model specifies a flow of control between tasks.

BusinessProcessMetric
Business process metrics are operational or structural measurements that track how the process is performing over time. Operational metrics deal directly with dynamic properties of business while structural metrics deal with static properties. E.g. Quantity measurements are a performance count or a measure of the amount of product produced by a single process case performance. Quality measurements are a determination of the value of the particular product in relation to some pre-determined quality norm. Time of performance is a measure of elapsed time between inception based on pre-condition and completion based on post-conditions being in place.
Tagged Values:
Metric. An OCL expression which defines the measurement.
startTrigger. An OCL expression which defines the condition which initiates the measurement.
stopTrigger An OCL expression which defines the condition which terminates the measurement.

BusinessTransactionUseCase
A business transaction use case is used to gather requirements for business transaction specifications.
Tagged Values:

requestingBusinessFunction. The business function that is implemented by the requesting business partner which is performing a role with respect to the use case e.g. procurement.
respondingBusinessFunction. The business function that is implemented by the responding business partner which is performing a role with respect to the use case e.g. fulfilment.

Lexicon
A lexicon is a repository of grammar components that are at its base, form a list or set of basic concepts or lexical entries (lexical affinity). It contains information about (a) the notation, (b) the semantics, (c)morphological properties, and (d) syntactic properties of its entries. The Lexicon must contain at least the idiosyncratic information about its entries. Any property of a concept or lexical entry that can be predicted by morphological or syntactic rule will be excluded from the Lexicon. Morphological rules or constructs may be identified by the Lexicon.

PartnerType
A partner type is an actor in a business collaboration use case. Partner types are manufacturer, distributor, retailer, end user, carrier and financier.

Agreement
An agreement is an arrangement between two partner types that specifies in advance the conditions under which they will trade (terms of shipment, terms of payment, collaboration protocols, etc.) An agreement does not imply specific economic commitments.
Tagged Values:
AgreementType. AgreementTypes classify and structure agreements. For example, an AgreementType might specify the kinds of terms and conditions that must be agreed upon for any instance of an agreement of the particular type. Examples of agreement types might include trading partner agreements and yearly economic contracts.

EconomicContract
A contract is subtype of agreement between partner types that some actual economic exchanges will occur in the future. Contracts can have recursive relationships with other contracts, for example, yearly contracts with monthly releases and weekly or daily shipping schedules. Contracts are containers for collections of commitments. For example, a purchase order is a contract wherein the line items are commitments.
Tagged Values:
initiateCondition. An economic contract term of effect is determined by the initiateCondition. This is an OCL constraint and may be defined by measurable elements such as a date, event or system metric.
terminactionCondition. An economic contract is no longer in effect if the terminationCondition has been true after the qualification of the iniateCondition. This is an OCL constraint and may be defined by measurable elements such as a date, event or system metric.

Economic Commitment
An economic commitment is an obligation to perform an economic event (that is, transfer ownership of a specified quantity of a specified economic resource type) at some future point in time. Order line items are examples of commitments.
Tagged Values:
measure. The measurement of an economic resource of the specified type to be transferred.
due. The condition that determines when the transfer of ownership is promised to occur. This is an OCL constraint and may be defined by elements such as a date, event or system metrics.

Reciprocity
Reciprocity is a mandatory relationship between two or more commitments. Business contracts require reciprocal commitments, called “consideration”.

EconomicResourceType
An economic resource type is the abstract classification or definition of an economic resource. For example, in an ERP system, ItemMaster or ProductMaster would represent the Economic Resource Type that abstractly defines an Inventory item or product. Forms of payment are also defined by economic resource types, e.g. currency.

EconomicResource
An economic resource is a quantity of something of value that is under the control of an enterprise, which is transferred from one partner type to another in economic events. Examples are cash, inventory, labor service and machine service.
Tagged Values:

measurement. The number and unit of the economic resource. Unit may be a unit of measure for products, a unit of time for services, or a currency for cash.

location. The location where the economic resource currently resides or is available.

BusinessEvent
A business event is a significant change in the state of one or more entities within a business, e.g. the taking of an order or a price change.

EconomicEvent
An economic event is the transfer of control of an economic resource from one partner type to another partner type. Examples would include sale, cash-payment, shipment, and lease.
Tagged Values:

measurement. The number and unit of the economic resource. that is being transferred.

Duality
Duality is a relationship between Economic Events, where one is the legal or economic consideration of the other. Examples include a payment for a product or service. Duality relationships occur between two or more economic events.

BTV concepts

BusinessAction
The state of a business transaction is defined by reciprocal Business Actions executed by an authorized role. This is an abstract class that is not a stereotype.
Tagged Values:

IsAuthorizationRequired. If a partner role needs authorization to request a business action or to respond to a business action then the sending partner role must sign the business document exchanged and the receiving partner role must validate this business control and approve the authorizer. A responding partner must signal an authorization exception if the sending partner role is not authorized to perform the business activity. A sending partner must send notification of failed authorization if a responding partner is not authorized to perform the responding business activity.
isNonRepudiationRequired. If non-repudiation of origin and content is required then the business
activity must store the business document in its original form for the duration mutually agreed to in a trading partner agreement. A responding partner must signal a business control exception if the sending partner role has not properly delivered their business document. A requesting partner must send notification of failed business control if a responding partner has not properly delivered their business document. This property provides the following audit controls: Verify sending role identity (authenticate) – Verify the identity of the sending role (employee or organization). For example, a driver's license or passport document with a picture is used to verify an individual's identity by comparing the individual against the picture. Verify content integrity – Verify the integrity of the original content sent from a partner role i.e. check that the content has not been altered by a 3rd party while the content was exchanged between partners.
timeToPerform. Both partners agree to perform a business transaction within a specific duration. A responding partner must exit the transaction if they are not able to respond to a business document request within the agreed timeout period. A sending partner must retry a business
transaction if necessary or must send notification of failed business control (possibly revoking a contractual offer) if a responding partner does not deliver their business document within the agreed time period. The time to perform is the duration from the time a business document
request is sent by a requesting partner role until the time a responding business document is "properly received" by the requesting partner role. Both partners agree that the business signal document or business action document specified as the document to return within the time to
perform is the "Acceptance Document" in an on-line offer/acceptance contract formation process.
TimeToAcknowlegeReceipt. Both partners agree to mutually verify receipt of a requesting business document within specific time duration. A responding partner must exit the transaction if they are not able to verify the proper receipt of a business document request within the agree timeout period. A sending partner must retry a business transaction if necessary or must send notification of failed business control (possibly revoking a contractual offer) if a responding partner does not verify properly receipt of a business document request within the agreed time period. The time to acknowledge receipt is the duration from the time a business document request is sent by a requesting partner until the time a verification of receipt is "properly received" by the requesting
business partner. This verification of receipt is an audit-able business signal and is instrumental in contractual obligation transfer during a contract formation process (e.g. offer/accept).
timeToAcknowledgeAcceptance. Both partners agree to the need for a business acceptance document to be returned by a responding partner after the requesting business document passes a set of business rules. The time to acknowledge business acceptance of a requesting business document is the duration from the time a requesting partner sends a business document until the time an acknowledgement of acceptance is "properly received" by the

requesting partner. A responding partner must exit the transaction if they are not able to acknowledge business acceptance of a business document request within the agreed timeout period. A sending partner
must retry a business transaction if necessary or must send notification of failed business control (possibly revoking a contractual offer) if a responding partner does not acknowledge acceptance of a business document within the agreed time period.

RequestingBusinessActivity
A requesting business activity is a business activity that is performed by a partner role requesting business service from another business partner role.
Tagged Values:

isNonRepudiationOfReceiptRequired. Both partners agree to mutually verify receipt of a requesting business document and that the receipt must be non-reputable. A receiving partner must send notification of failed business control (possibly revoking a contractual offer) if a responding partner has not properly delivered their business document. Non-repudiation of receipt provides the following audit controls.
Verify responding role identity (authenticate) – Verify the identity of the responding role (individual or organization) that received the requesting business document.
Verify content integrity – Verify the integrity of the original content of the business document request.
retryCount. Both partners agree to the number of times to retry a transaction when a time-out-exception condition is signaled. This parameter only applies to time-out signals and not business process controls or
document content exceptions.

RespondingBusinessActivity
A responding business activity is a business activity that is performed by a partner role responding to another business partner role's request for business service.
Tagged Values:

isIntelligibleCheckRequired. Both partners agree that a responding partner role must check that a
requesting document is not garbled (unreadable, unintelligible) before verification of proper receipt is returned to the requesting partner. Verification of receipt must be returned when a document is "accessible" but it is preferable to also check for garbled transmissions at the
same time in a point-to-point synchronous business network where partners interact without going through an asynchronous service provider.

InformationEntity
An information entity realizes structured business information that is exchanged by partner roles performing activities in a business transaction. Information entities include or reference other information entities through associations. A secure information entity is an information entity with security controls. Security controls must be specified when information must be secured within an enterprise until it is accessed by an authorized partner role. These parameters on this model element must be specified in a
manner that ensures document integrity by maintaining a "chain-of-custody" from the sender to the intended recipient of the business information.
Tagged Values:

isConfidential. The information entity is encrypted so that unauthorized parties cannot view the
information.
isTamperProof. The information entity has an encrypted message digest that can be used to check
if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.

isAuthenticated. There is a digital certificate associated with the document entity. This provides
proof of the signer's identity.

StructuredDocument
A structured document is an information entity container.

UnstructuredDocument
An unstructured document is any document that is not comprised of document entities.
Tagged Values:

dataType. This property specifies the document type. It is recommended that a registered MIME
type by used for this property (refer to http://www.iana.org) for registered MIME types. Partners can agree to use their own experimental MIME types.

OrganizationalRole
Only an organization performs a particular role in an e-business collaboration. An employee does not perform these activities.

AuthorizedRole
A partner role is a functional role, an employee role or an organizational role. Either an employee role or an organizational role can perform a functional role. An organizational role must be performed by a conforming business service.

EmployeeRole
An employee for business/legal reasons can only perform an employee role. Usually the details of the employee must be captured and stored/transmitted to another partner for auditing/liability purposes when the two partner roles are not in the same organization.

BusinessTransaction
A business transaction is a set of business information and business signal exchanges between two business partners that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded. Business transactions can be formal as in the formation of on-line offer/acceptance business contracts and informal as in the distribution of product announcements. Business transactions can be comprised of sub-transactions.
Tagged Values:

isSecureTransportRequired. Both partners must agree to exchange business information using a secure transport channel. The following security controls ensure that business document content is protected against unauthorized disclosure or modification and that business services are protected against unauthorized access. This is a point-to-point security requirement. Note that this requirement does not protect business information once it is off the network and inside an enterprise. The following are requirements for secure transport channels. Authenticate sending role identity – Verify the identity of the sending role (employee or organization) that is initiating
the role interaction (authenticate). For example, a driver's license or passport document with a picture is used to verify an individual's identity by comparing the individual against the picture.
Authenticate receiving role identity – Verify the identity of the receiving role (employee or organization) that is receiving the role interaction. Verify content integrity – Verify the integrity of the content exchanged during the role interaction i.e. check that the content has not been altered by a 3rd party. Maintain content confidentiality – Confidentiality ensures that only the intended, receiving role can read the content of the role interaction. Information exchanged during role interaction must be encrypted when sent and decrypted when received. For example, you seal envelopes so that only the recipient can read the content.

BusinessCollaborationProtocol
A business collaboration protocol choreographs one or more business transaction activities. A business collaboration protocol is not a transaction and should be used in cases where transaction rollback is inappropriate. For example, a buying partner may request a purchase order by a selling partner. The selling partner may partially accept the purchase order and thus complete the transaction but may only return shipping information on part of the order. The buying partner is sent any number of later notifications regarding the outstanding portions of the order until the order is completely reconciled.

BusinessPartner
The business partners that participate in business collaborations are enumerated for each business collaboration protocol. Partners provide the initiating and responding roles in the protocol.

BusinessTransactionActivity
A business transaction activity is a business collaboration protocol activity that executes a specified business transaction. The business transaction activity can be executed more than once if the *isConcurrent* property is true.
Tagged Values:

timeToPerform. Both partners agree to perform a business transaction activity within a specific duration. The initiating partner must send a failure notification to a responding partner on timeout. A responding partner simple terminates its activity. The time to perform is the duration from the time a business transaction activity initiates the first business transaction until there is a transition back to the initiating business transaction activity. Both partners agree that the business signal document or business action document specified as the document to return within the time to
perform is the "Acceptance Document" in an on-line offer/acceptance contract formation process.

isConcurrent. If the business transaction activity is concurrent then more than one business transaction can be open at one time. If the business transaction activity is not concurrent then only one business transaction activity can be open at one time.

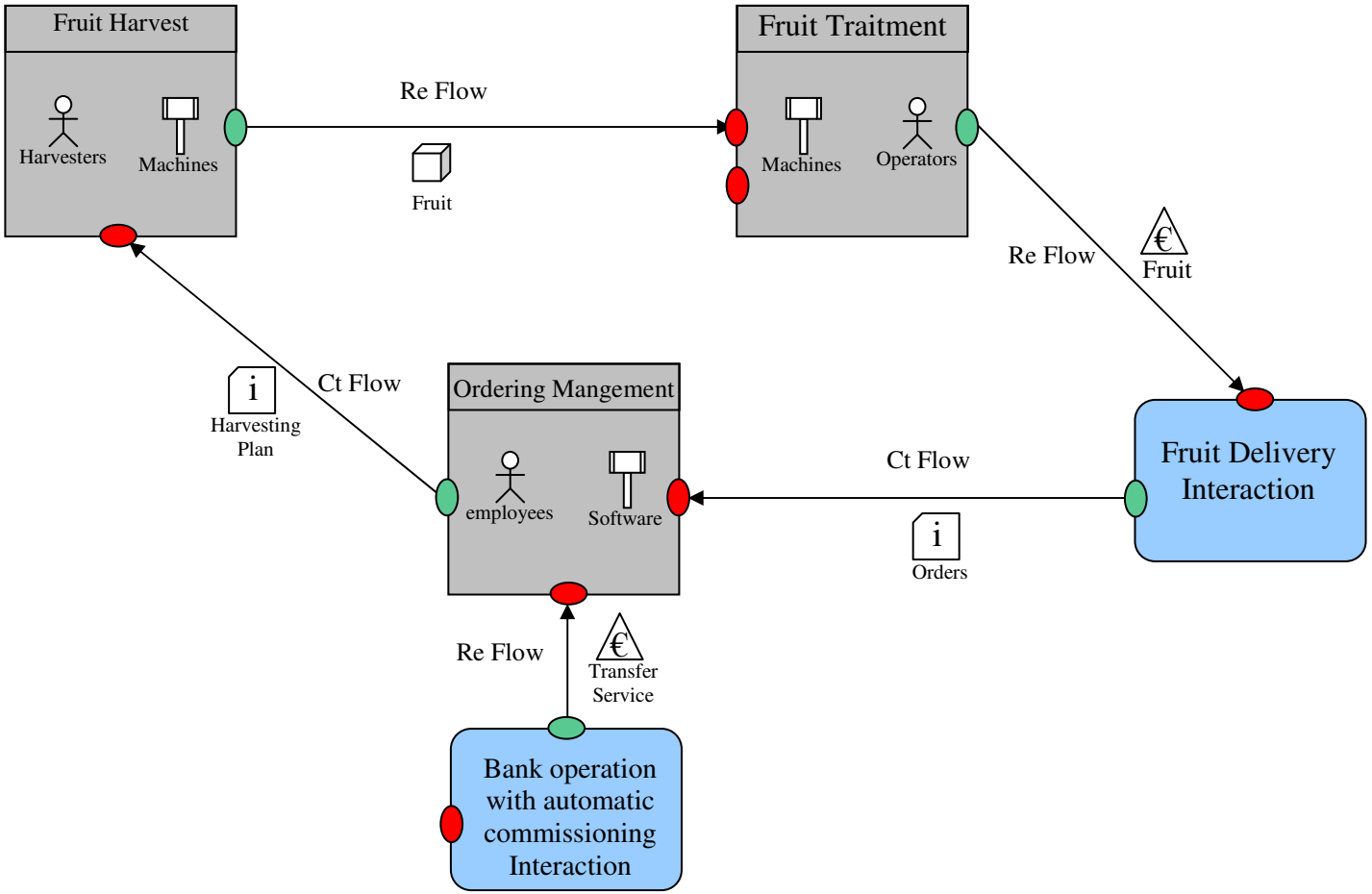
DocumentEnvelope
A document envelope is a container for structured and unstructured business documents.

Business Transaction Activity Model Elements
Business Transaction Activity elements are specialized elements derived from a RequestingBusinessActivity element. Each element defines as a stereotype, the default value for each required tags in support of each Business Transaction pattern.

Annex 3. EBCML case study: other models

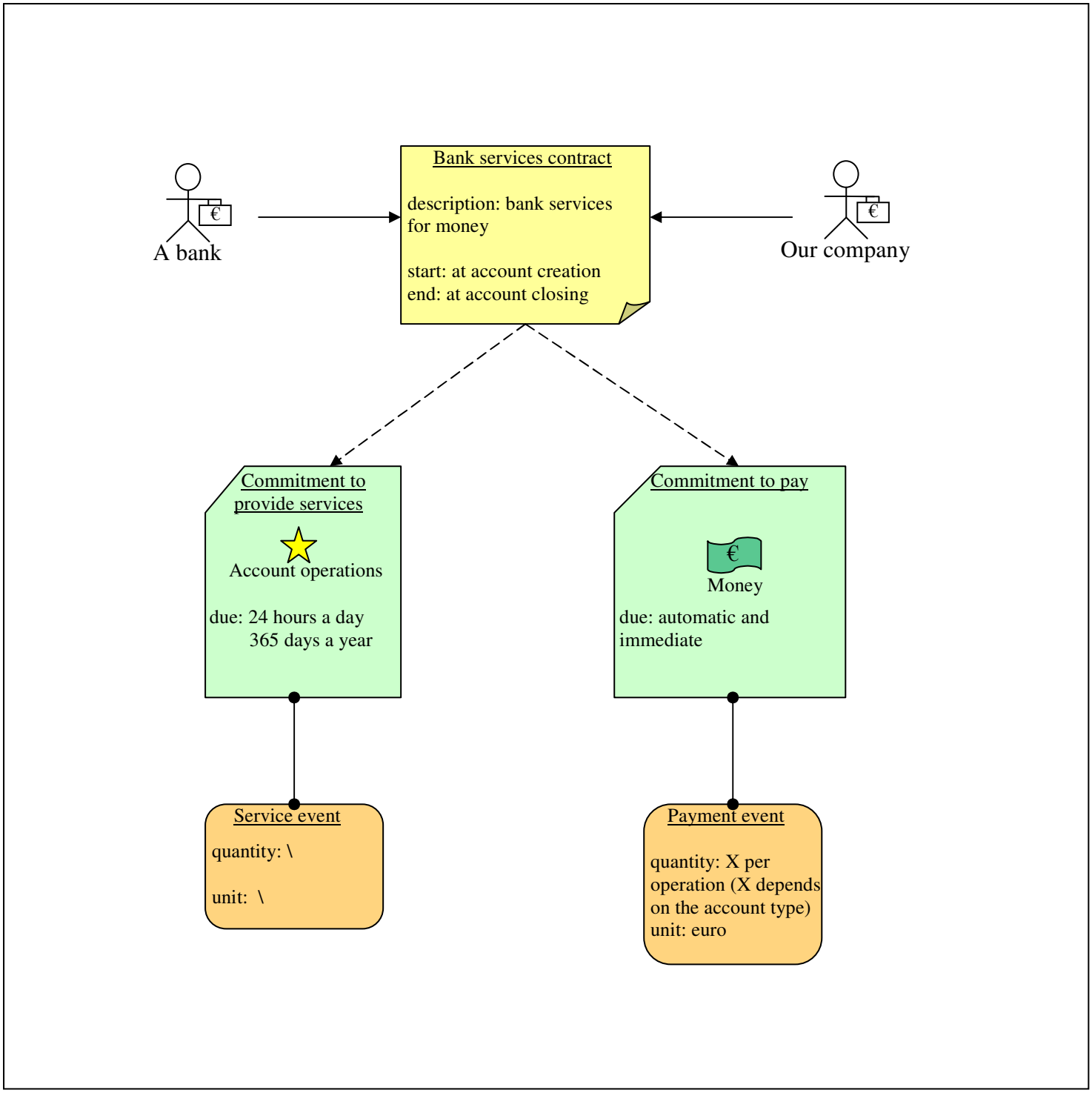
BigFruit

Enterprise View

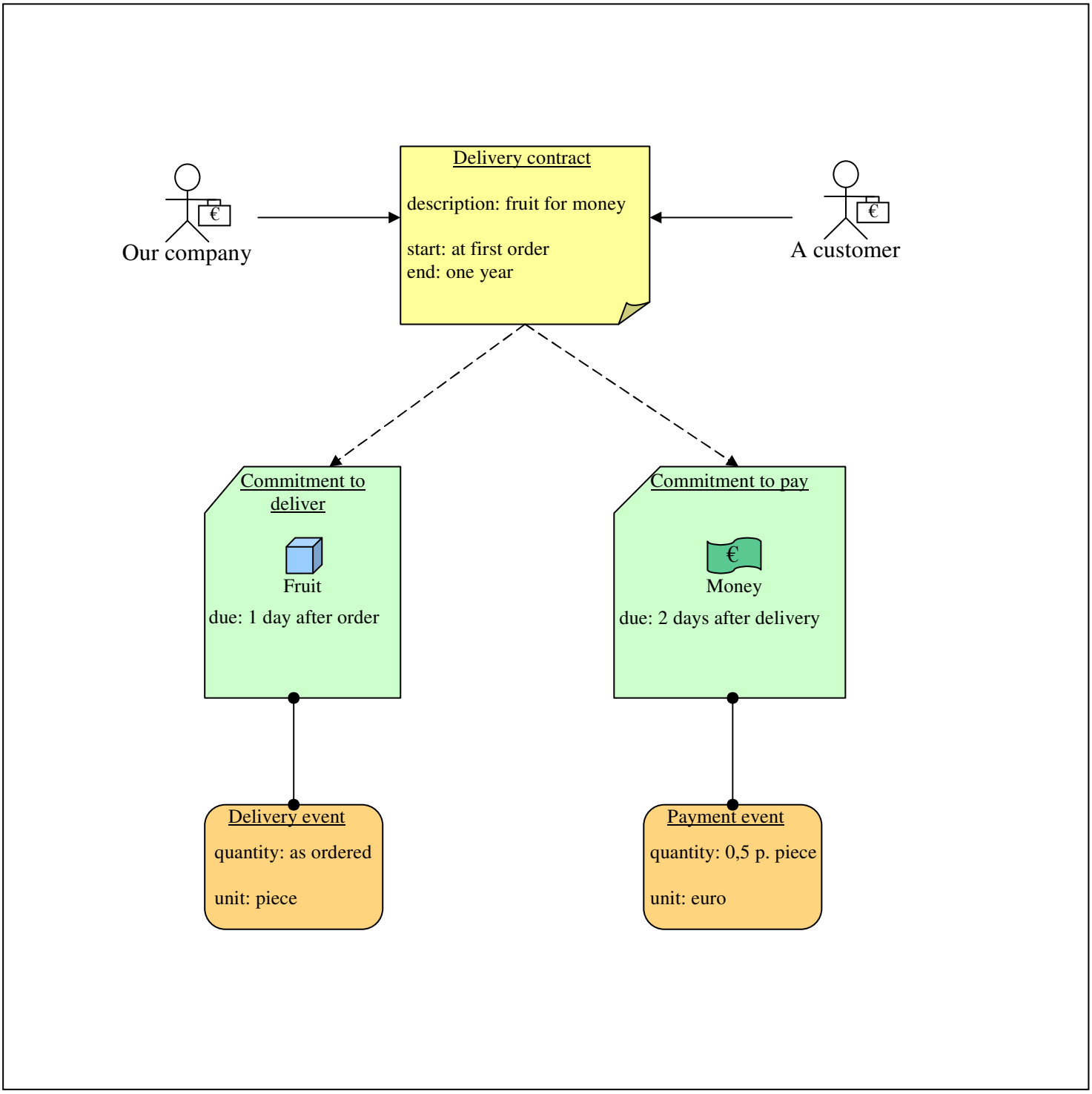


Collaboration View

Bank Account Collaboration

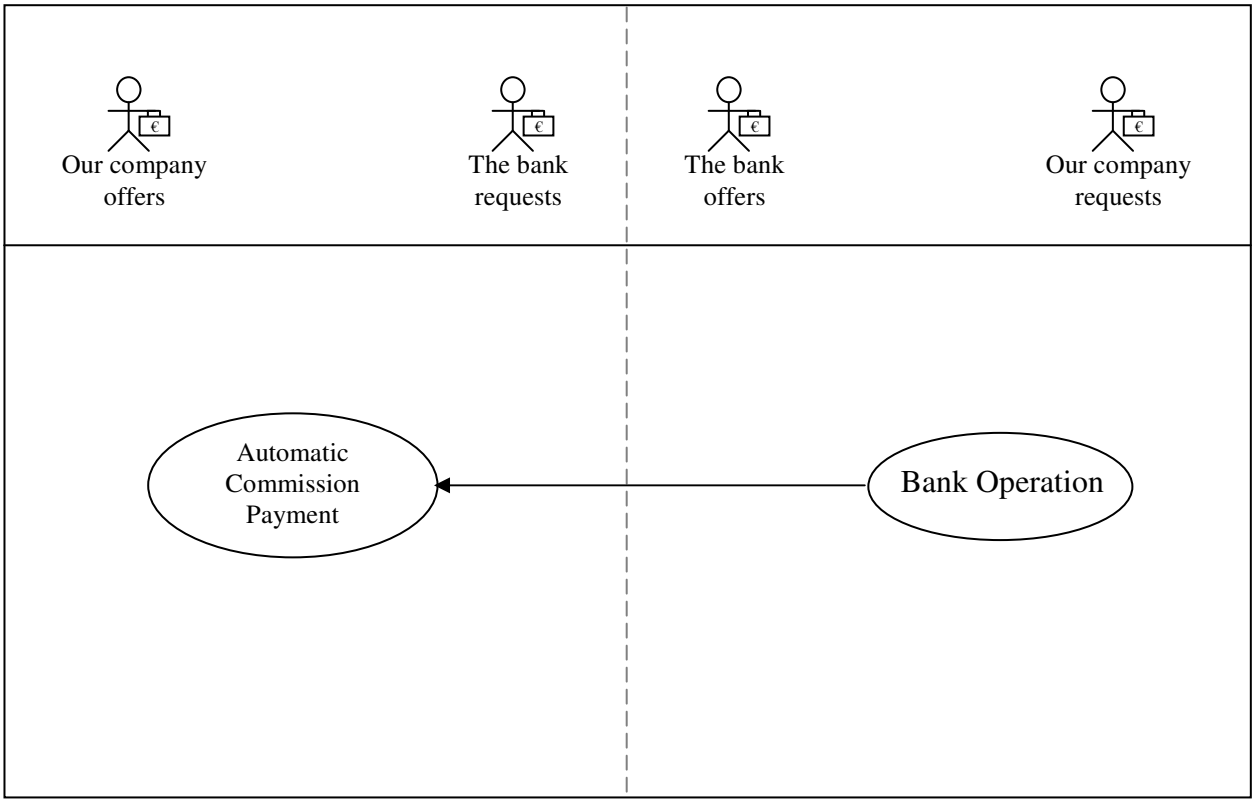


Fruit Delivery Collaboration

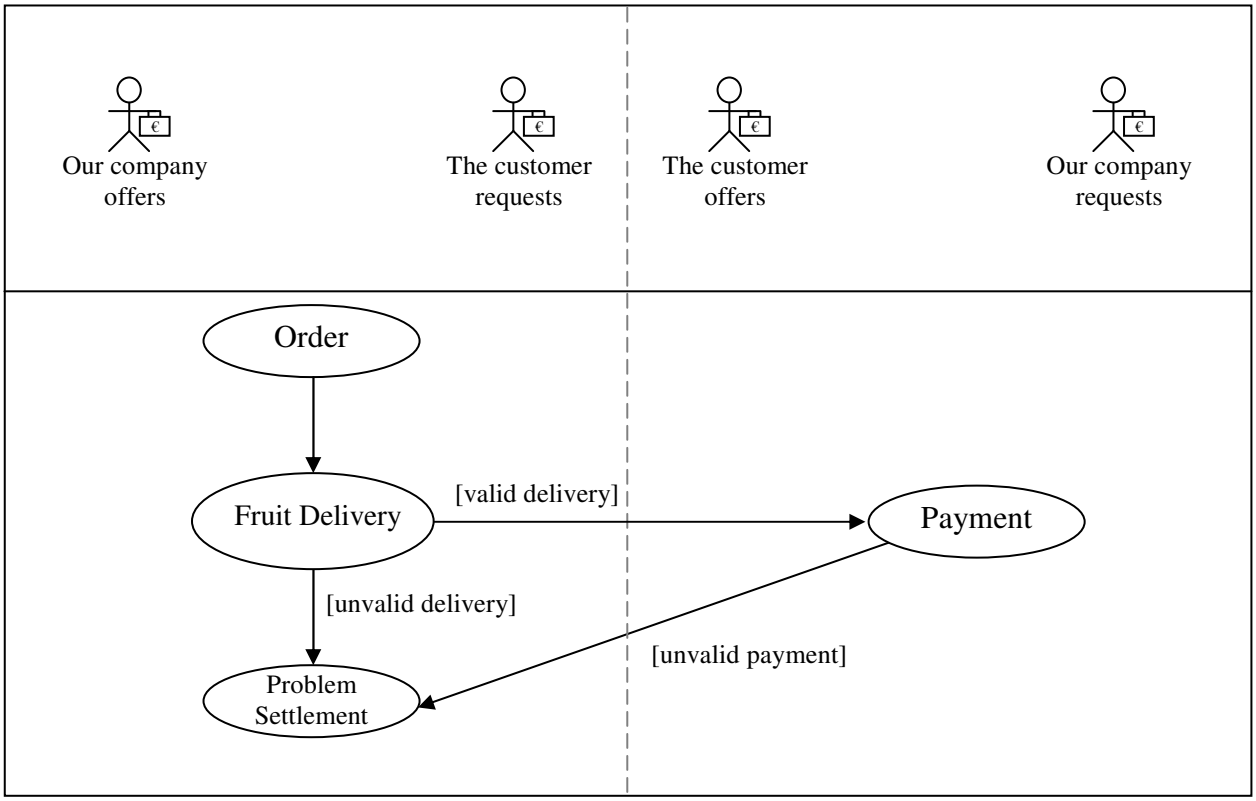


Interaction View

Bank Account Collaboration



Fruit Delivery Collaboration

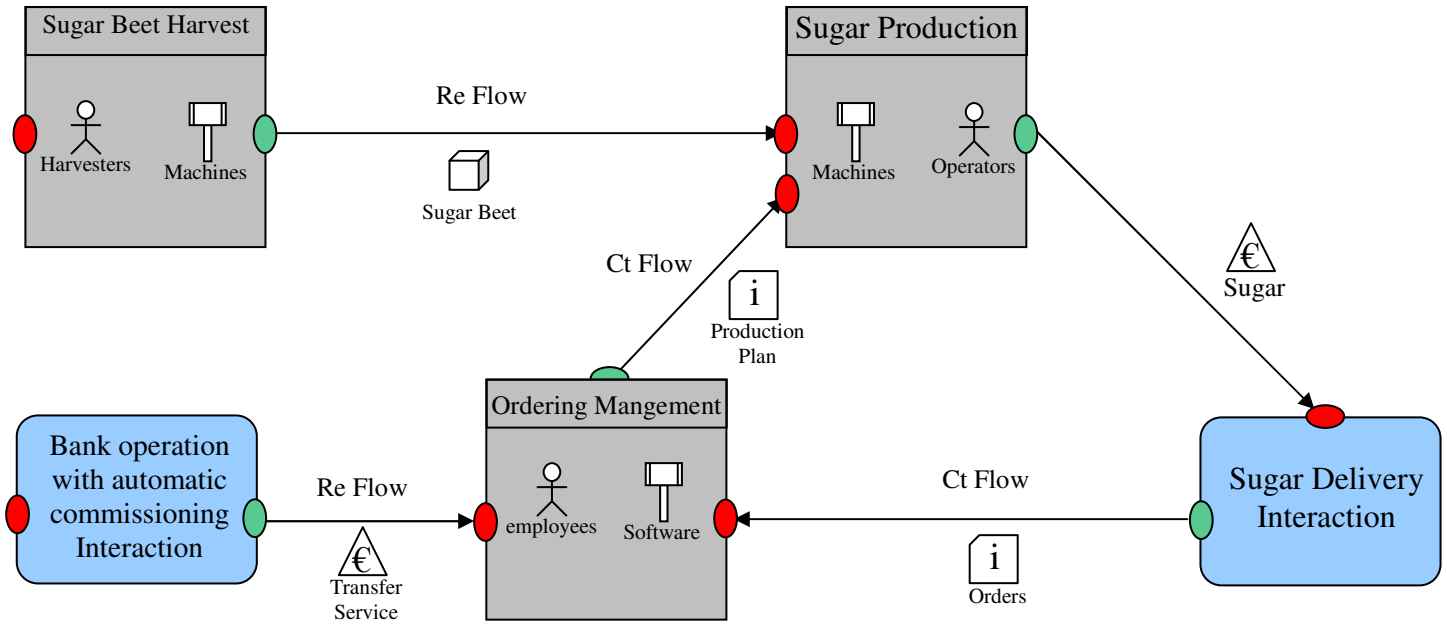


Information Exchange View

The information exchanges in which BigFruit participates are nearly identical to those described in CoolDrinks' Information Exchange View. It is therefore useless to present them here.

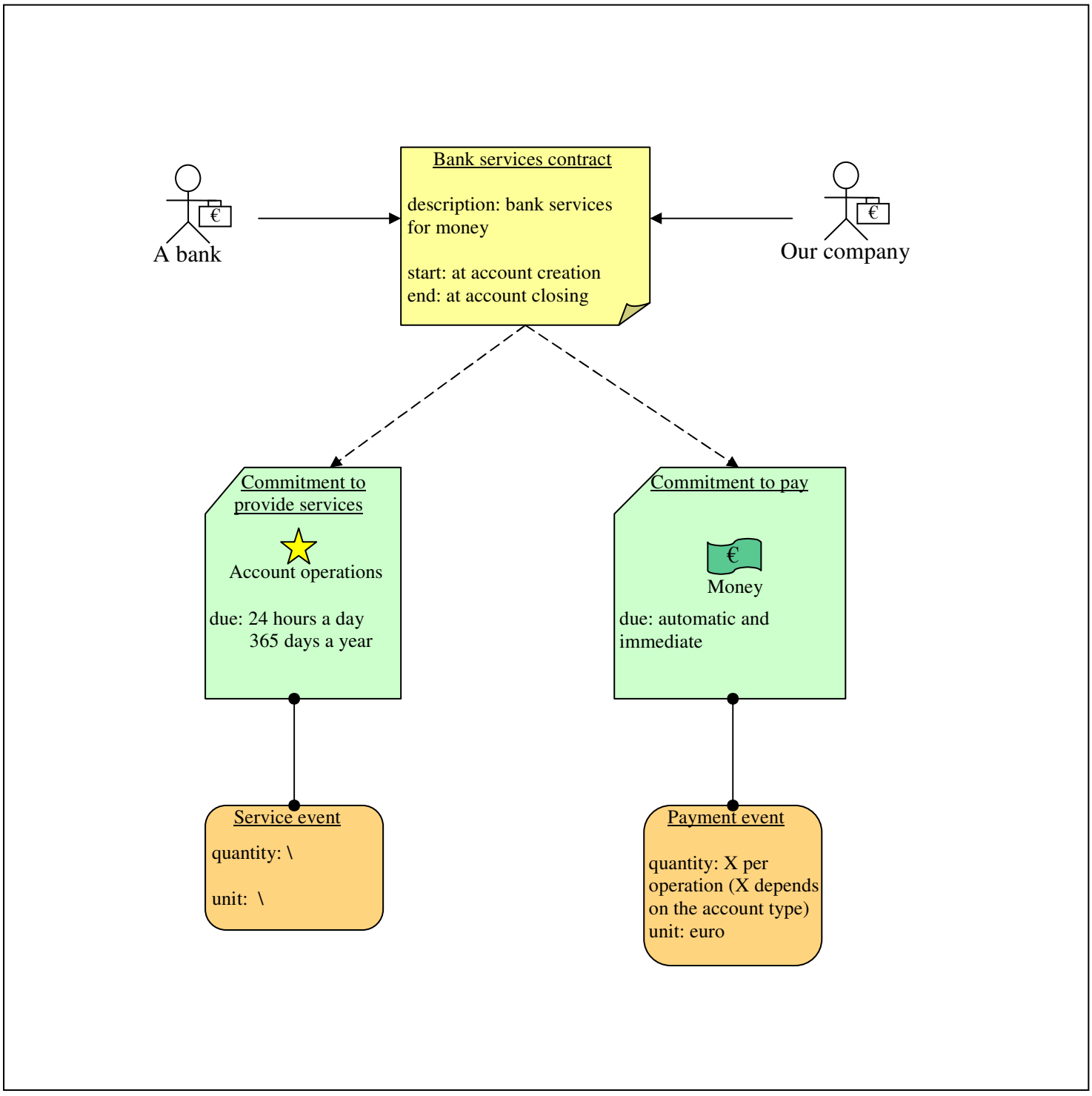
SuperSugar

Enterprise View

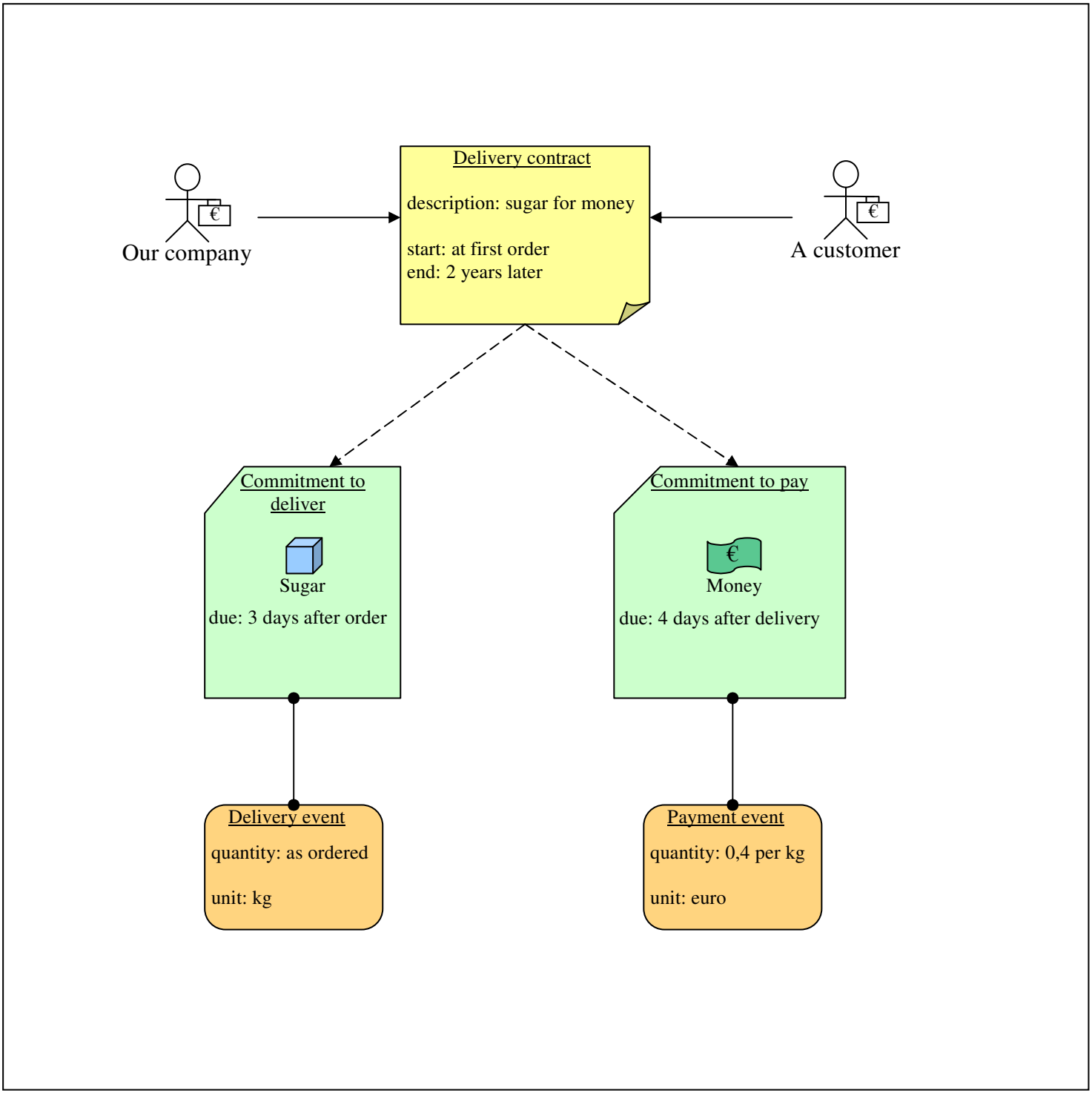


Collaboration View

Bank Account Collaboration

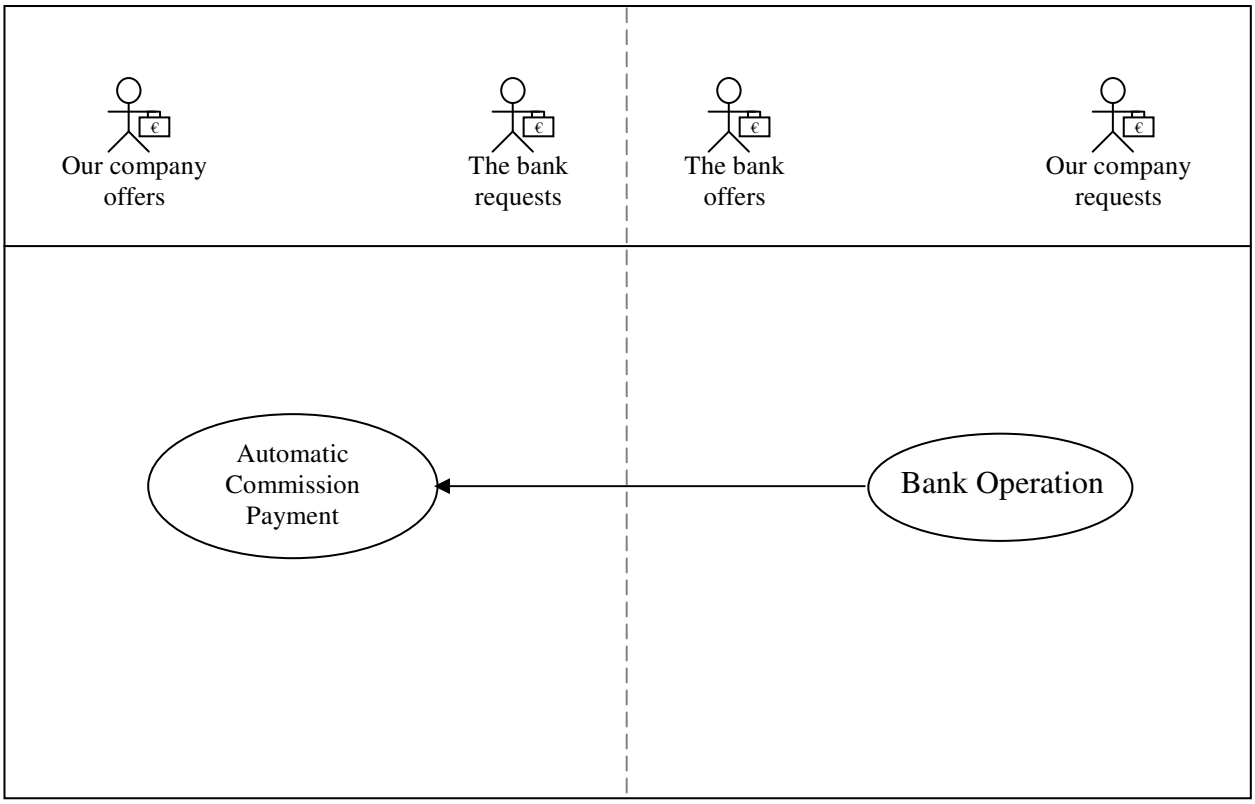


Sugar Delivery Collaboration

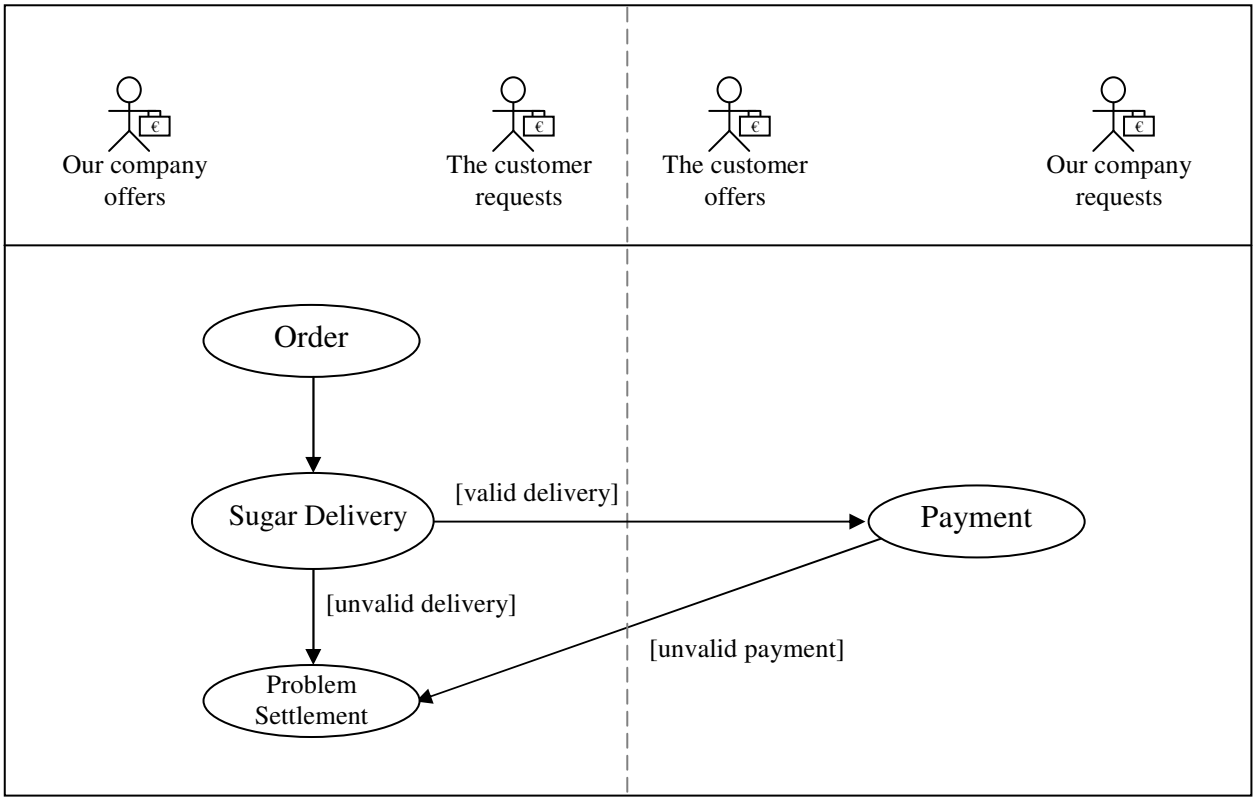


Interaction View

Bank Account Collaboration



Sugar Delivery Collaboration

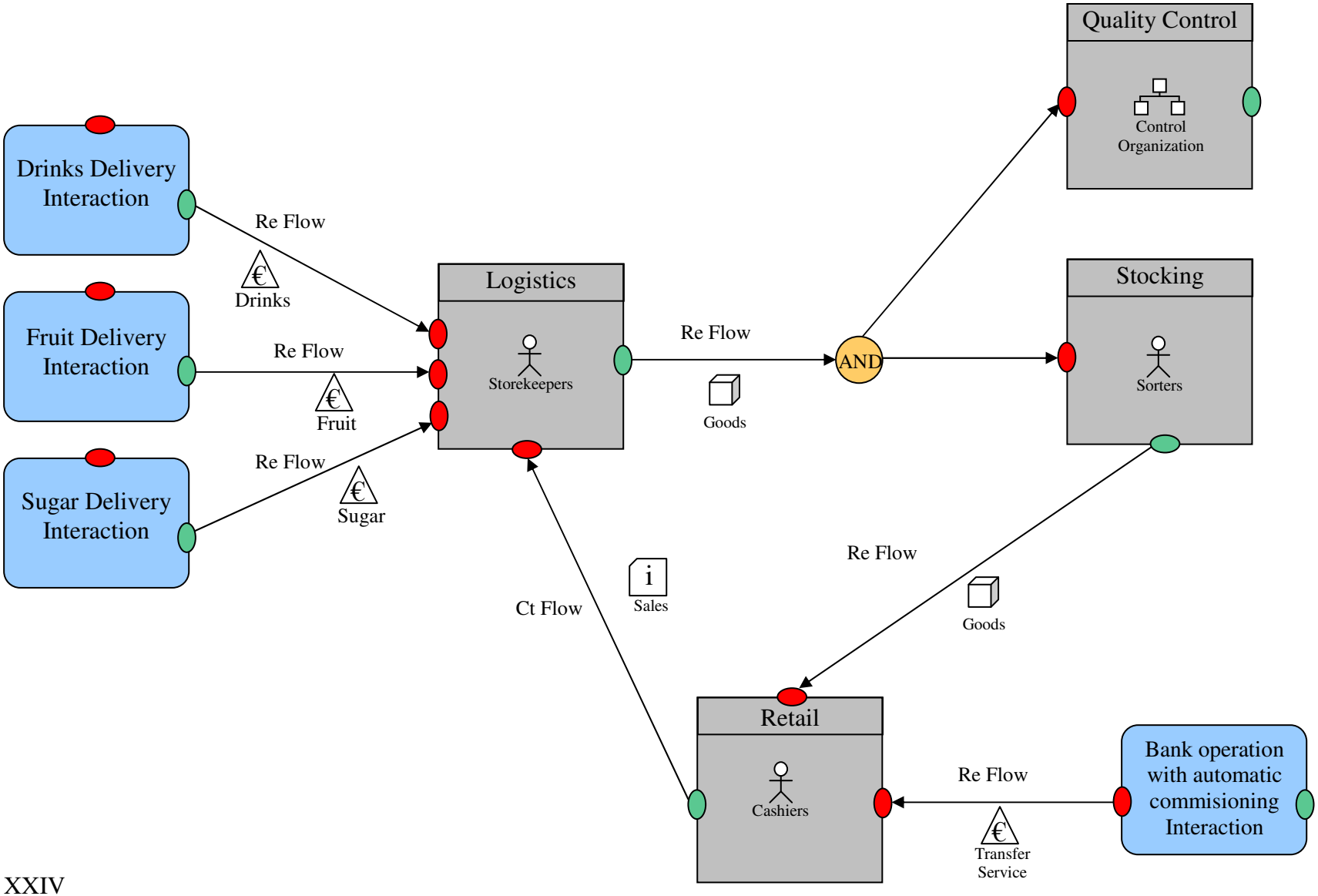


Information Exchange View

The information exchanges in which SuperSugar participates are nearly identical to those described in CoolDrinks' Information Exchange View. It is therefore useless to present them here.

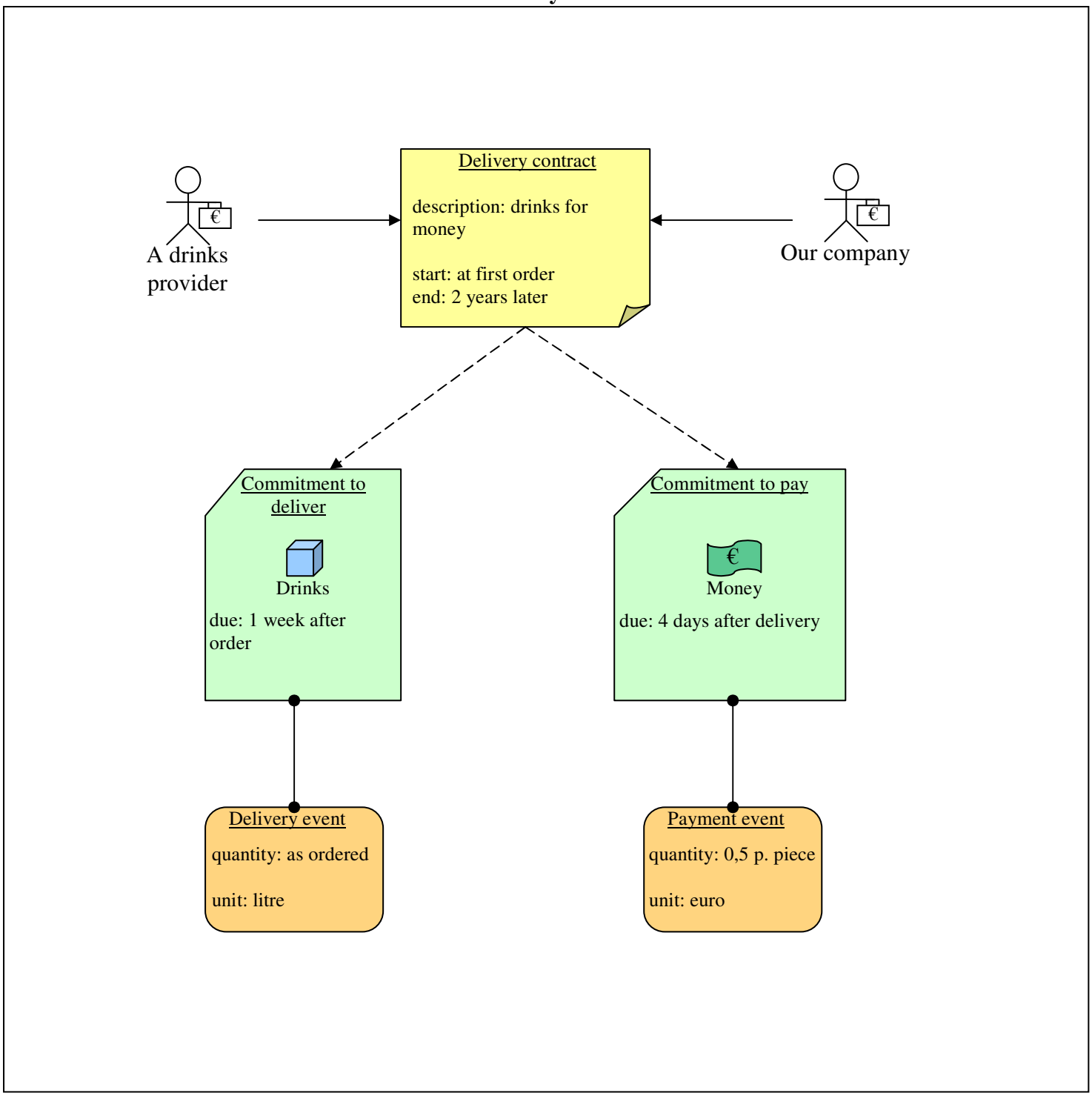
StarMarket

Enterprise View

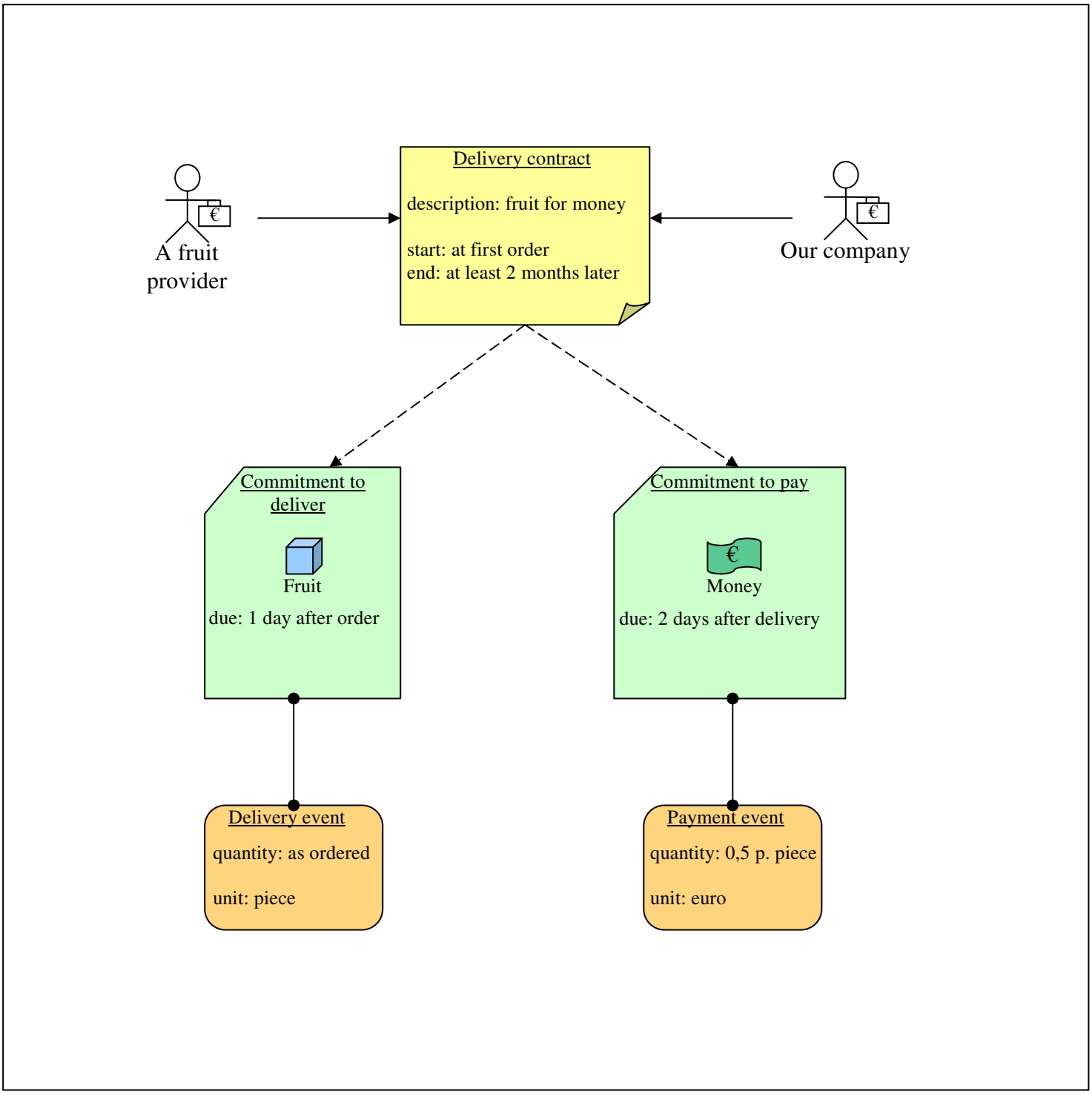


Collaboration View

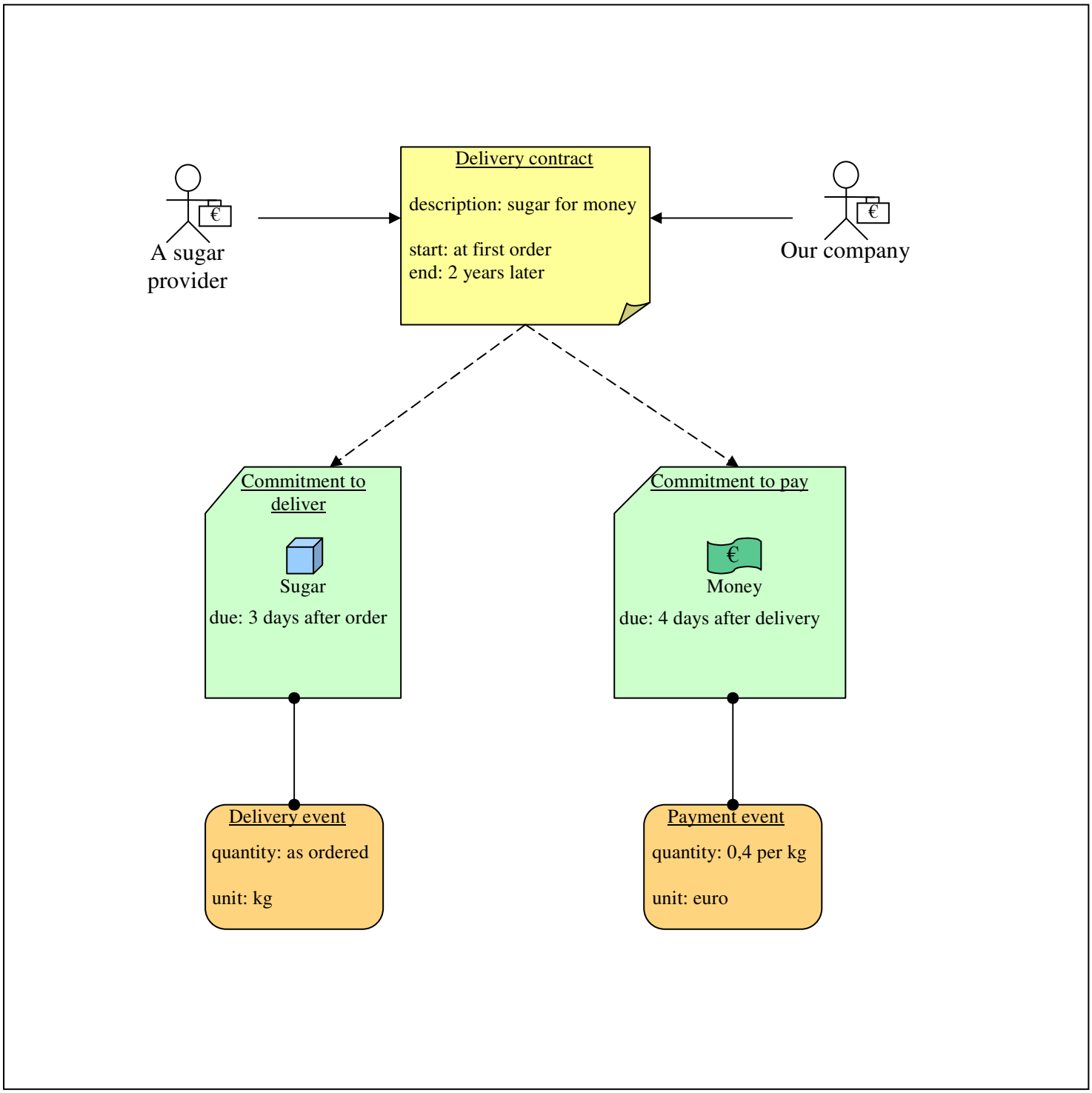
Drinks Delivery Collaboration



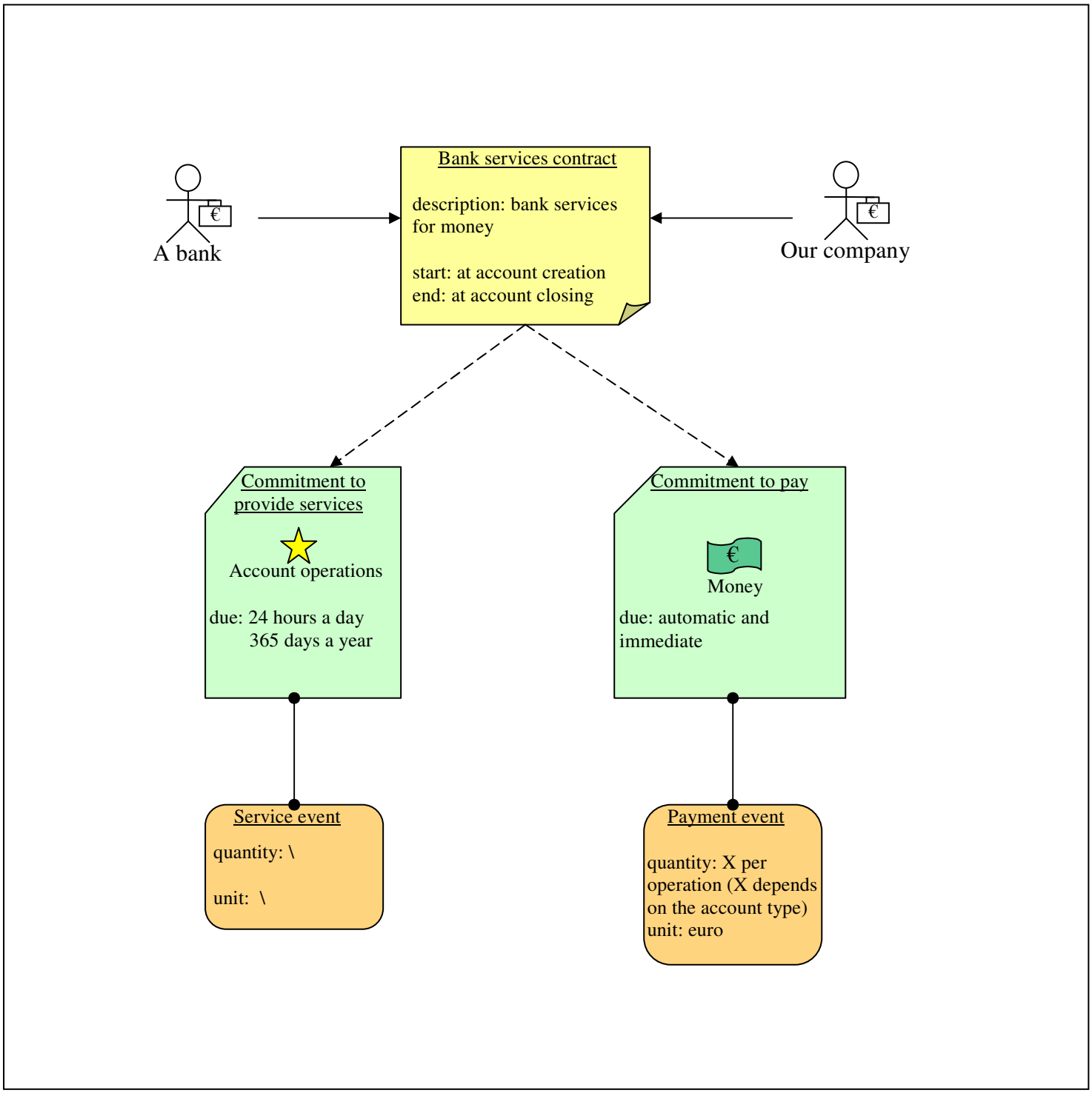
Fruit Delivery Collaboration



Sugar Delivery Collaboration

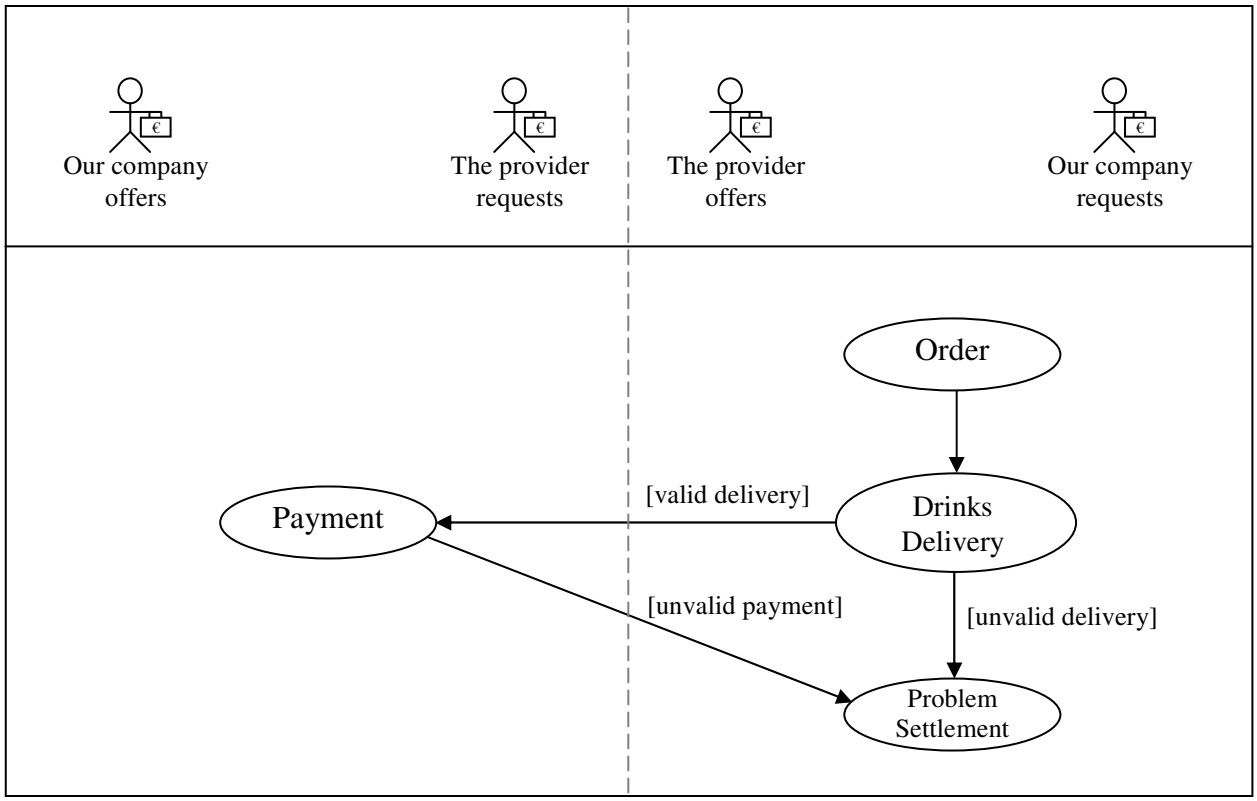


Bank Account Collaboration

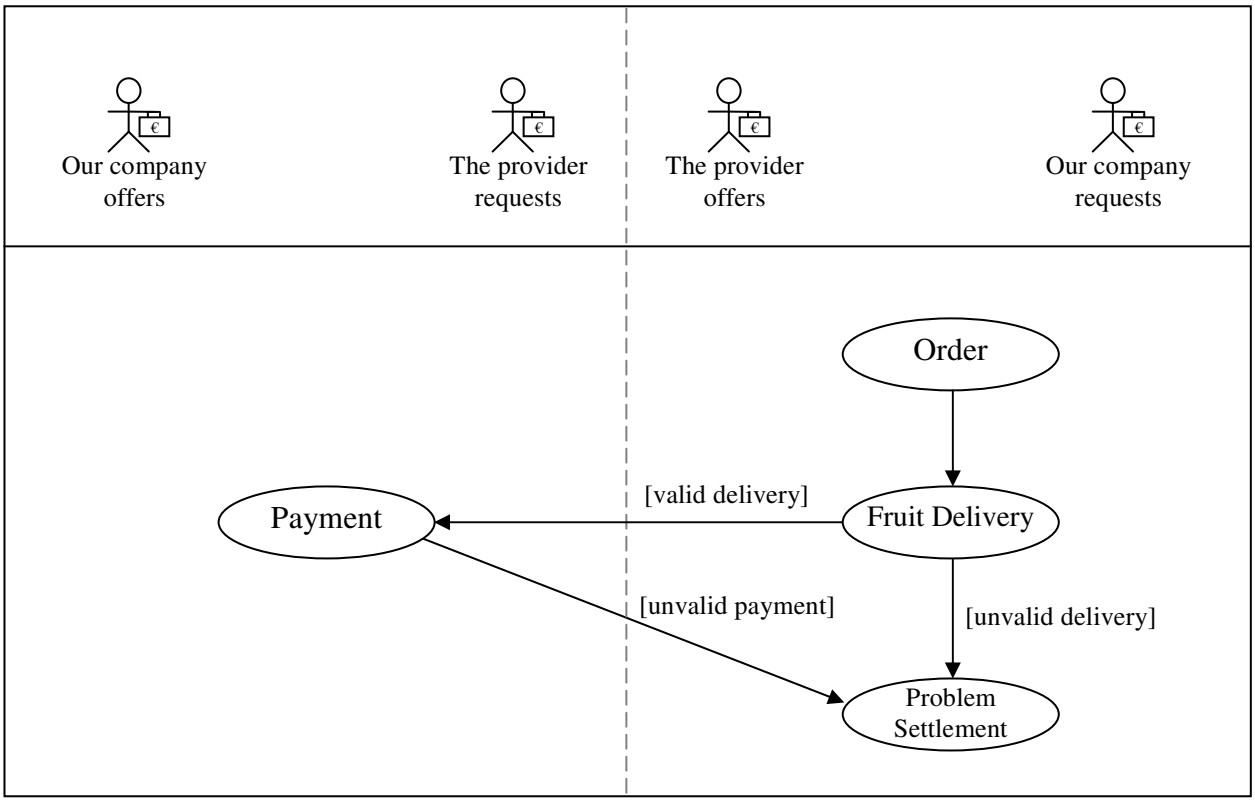


Interaction View

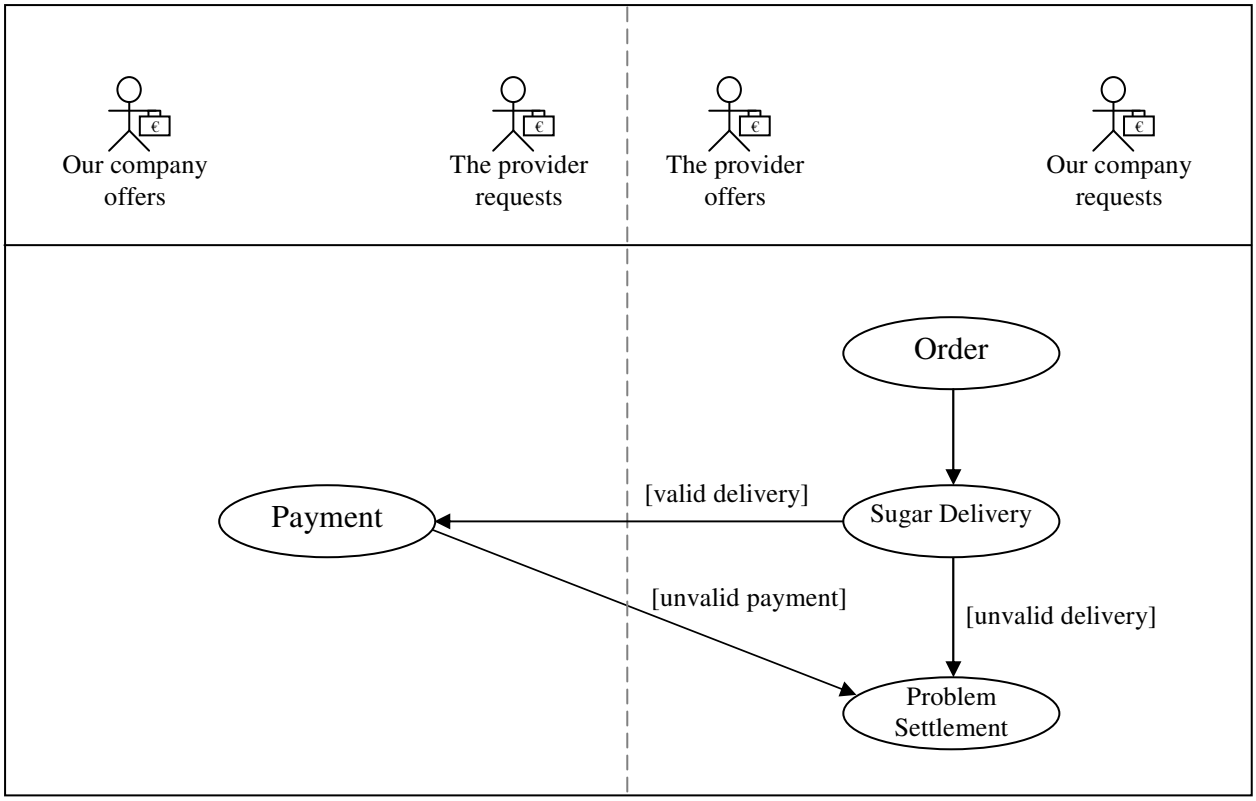
Drinks Delivery Collaboration



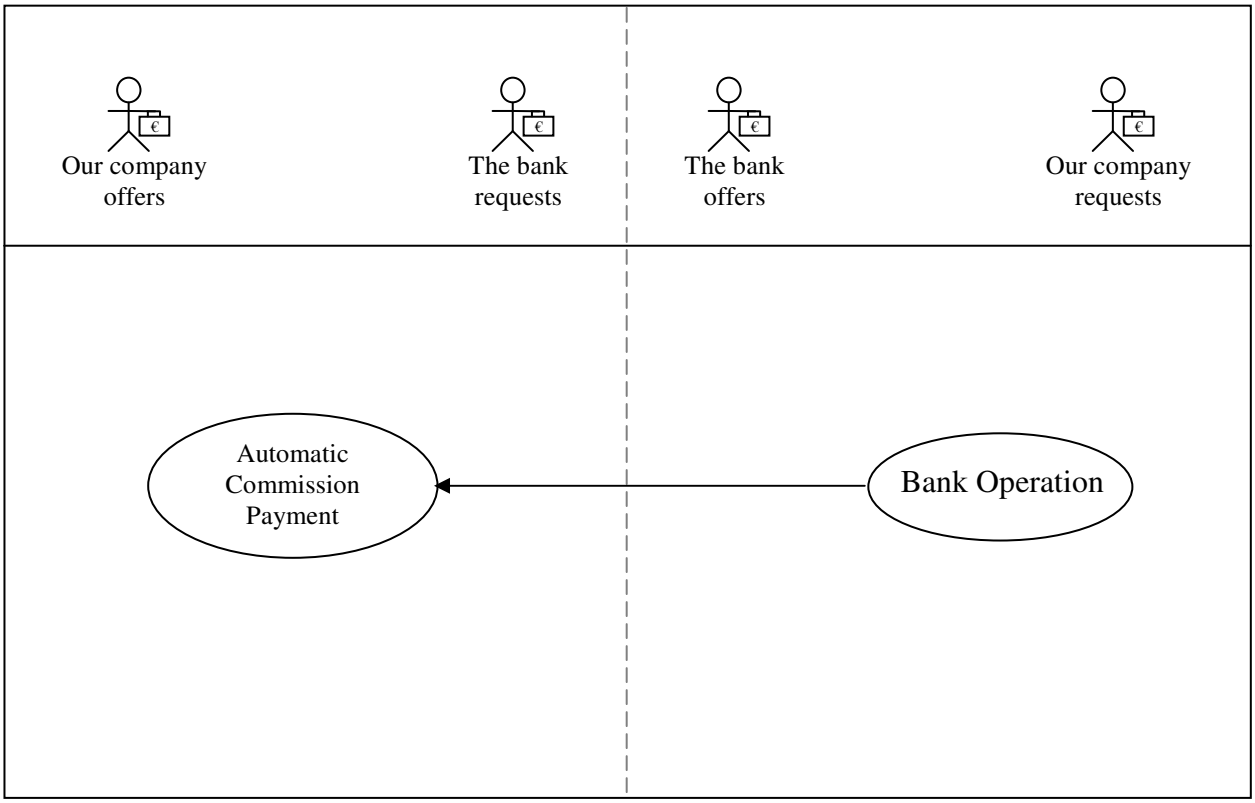
Fruit Delivery Collaboration



Sugar Delivery Collaboration



Bank Account Collaboration

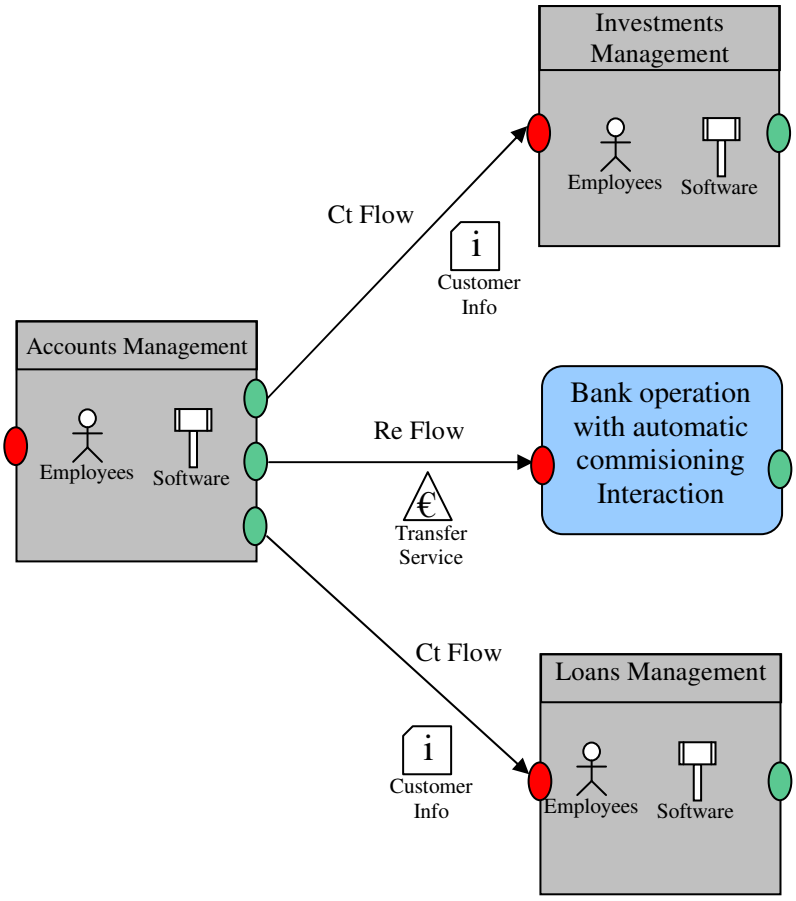


Information Exchange View

The information exchanges in which StarMarket participates are nearly identical to those described in CoolDrinks' Information Exchange View. It is therefore useless to present them here.

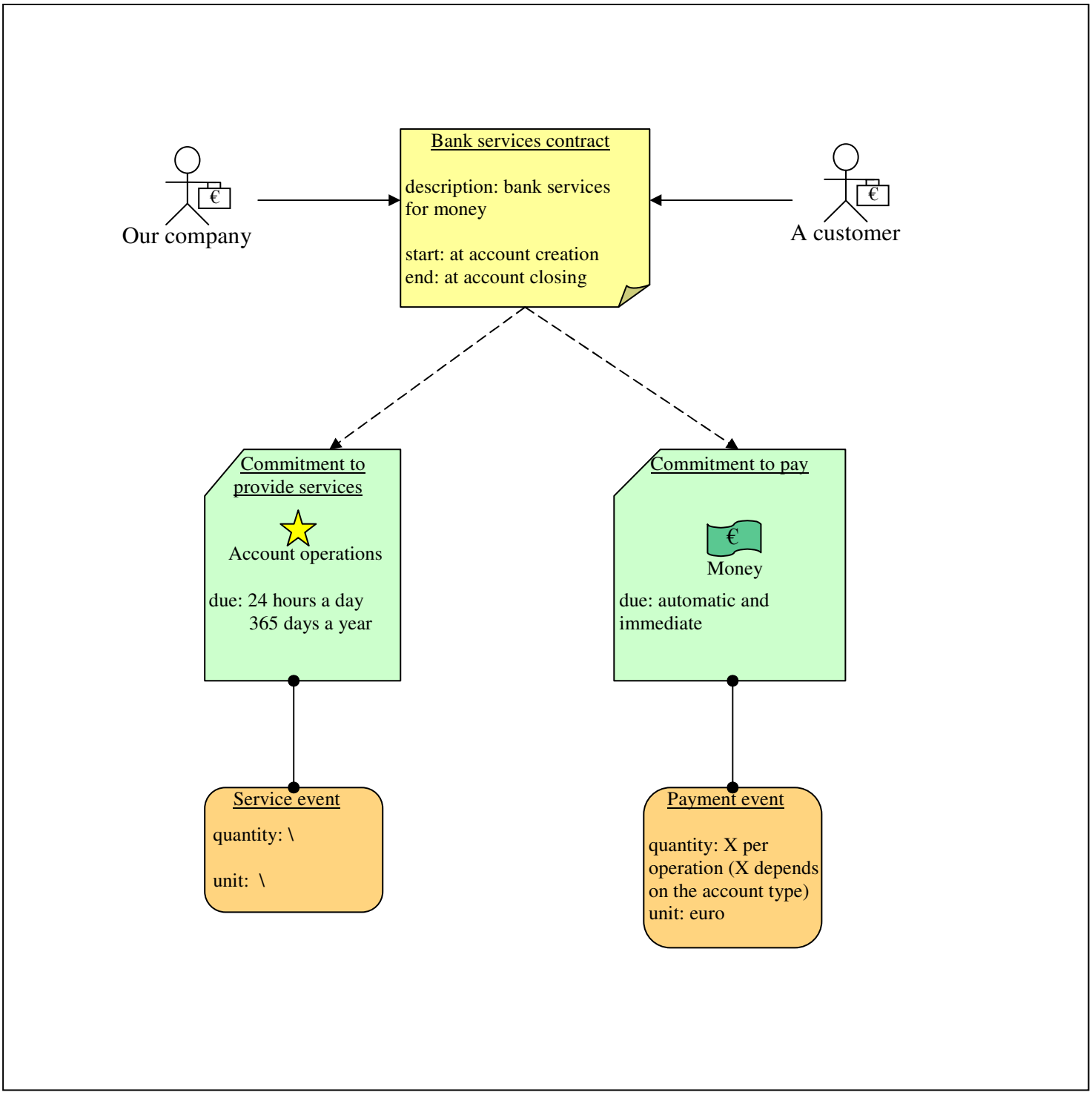
UniversalBank

Enterprise View



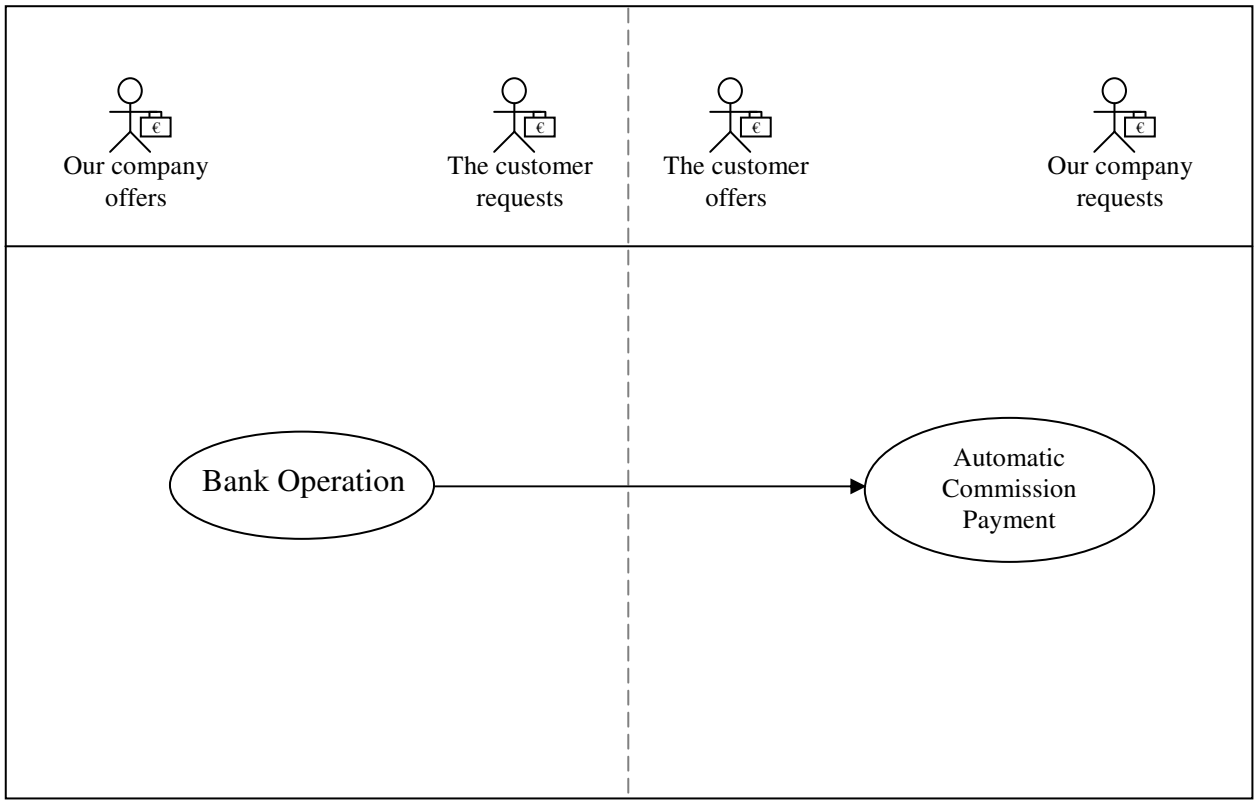
Collaboration View

Bank Account Collaboration



Interaction View

Bank Account Collaboration



Information Exchange View

Interaction "Bank Operation"

Our system	Our customer
	1. Connect
2. Ask for <Customer ID>	
	3. Send <Customer ID>
4. Ask for <Bank Operation Information>	
	5. Send <Bank Operation Information>
6. Execute operation	
	7. Disconnect